

# 第 1 章 绪论

## 1.1 人脑与人的智能、人工神经网络与智能机器、人工智能与计算智能

现代人的大脑重量平均值约为 1400g 大脑内约含 1000 亿(  $10^{11}$  )个神经元 每个神经元与其他神经元之间约有 1000 个连接 这样 大脑内约有  $10^6$  亿(  $10^{14}$  )个连接。人的智能行为就是由如此高度复杂的组织所产生的。浩瀚的宇宙中,就复杂度而言,可能只有包含数千亿颗星球的银河堪与人脑相比拟。因此人的智能行为是如此复杂也就毫不足奇了。

人脑的基本组成单位——神经元(neuron)——是一种特化的细胞,其示意结构如图 1-1 所示。神经元表面有许多为引进输入信号的短突起,称为树突(dendrite)而输出信号的是一根长突起,称为轴突(axon),轴突周围包有髓鞘。树突的任何部位都可以与来自其他神经元的轴突末梢建立联系,构成突触(synapse)。神经元的活动和神经元之间信号的传递是一个极复杂的生物化学——电过程。概言之,一个神经元只能取两种状态——激发态(兴奋状态)或抑制态,当神经元从抑制态转变为激发态时,即沿自身的轴突送出一个脉冲信号,此信号通过突触送到其他神经元。突触可分为兴奋型和抑制型两种且可能具有不同的强度,这就是说虽然神经元送出的脉冲信号是单一的,但随着突触的不同,接收方收到的是强度各异的兴奋或抑制信号。当一个处于抑制态的神经元接收到的信号总和为兴奋性且超过某一阈值时,即转变为兴奋态,且在传出自身的电信号以后逐渐减退为抑制态,在一短段时间中此神经元对外来信号不再作出响应(这段时期称为不应期)。在 1.5 节中还要对此机理作阐述。此外,大脑的工作具有模块化和层次化的特点,

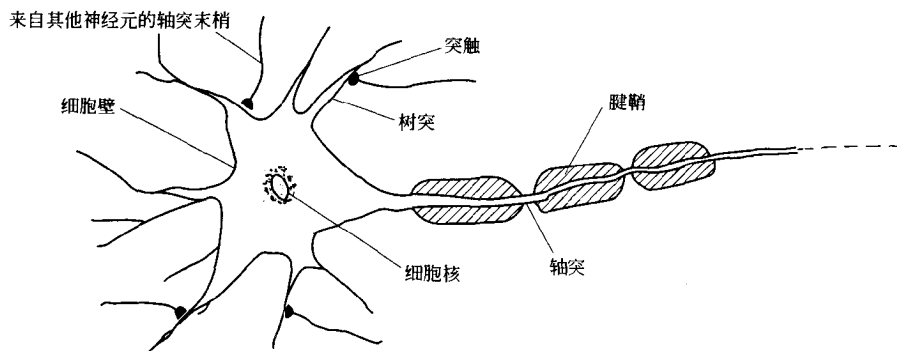


图 1-1 神经元结构示意图

即大脑皮层分成若干功能区，某个区域中互相协同作用的一群神经元组成具有某种特定功能的模块，例如对视觉信号作初步加工——检测出一帧图像中的边缘、线段及其方向等。对理解一帧图像而言，除了这种低层次的信息加工模块外，还有高层和更高层的模块对来自低层模块的信息作进一步加工（模块之间的信息交换是双向的，即既有“由底向上”的又有“由顶向下”的信息传送）。

什么是人的智能呢？对智能作出严格定义相当困难。下面只列出人的智能活动所涉及的一些领域，所列的这些智能行为之间往往有紧密关系，很难严格区分。此外，所列出的项目是极不完备的且每个项目还能分成若干更细的子项目，而且还可能涉及深层次的心理学、哲学和认知科学等问题。这里做的只是粗略介绍。

### 1. 感知和认知

感知是大脑通过各种感觉器官接受外界的声、光、触、嗅等信息。初级的认知是以感知信息为基础，辨识出不同事物并建立各特定事物及其特殊属性的概念。例如，每一头具体的牛都与另一头不同，但是可以建立关于牛这种动物的概念。这个概念包含世界上所有的牛和它们的共同属性，当然还可能包含更多深层次的信息。更高级的认知涉及各事物之间的关系以及抽象的概念，例如 5 这个数字可用于 5 头牛也可用于 5 个苹果。

### 2. 记忆

大脑不可能将所有接收到的信息全部存储起来，记忆存储与信息的加工（认知）和选择（集中注意、兴趣等）相关。记忆的另一面是提取出所存的某一部分信息，这种信息的搜索和提取往往是联想式的。这就是在存储各种信息时必然还需存储它们之间的联系，而不是存储孤立的信息，这样才能进行联想式的记忆项搜索提取。记忆是大脑最重要的功能之一，它是其他智能活动的基础。

### 3. 学习与知识

学习是大脑最重要的功能之一。学习的作用是获取、存储（记忆）和使用知识。知识是一种结构性的信息，例如因果关系（知因求果或反过来知果求因）、空间关系、时间关系、推理关系以及根据事物的特征进行分类和模式识别等等。无论何种知识，大体上都可以表现为一个称为输入的变向量对一个称为输出的变向量之间的映射关系。例如，前者是因后者是果；前者是一个函数的变量后者是函数值；前者是一个类别或模式的特征向量后者是分类或模式识别的结果，等等。需要说明的是，分类的作用是根据某种事物的特征来判断其属于哪一种具有特定意义的类别，例如当一个人体细胞只有具备了某些特征后就被认为变成了癌细胞，反之则否。从这个意义而言，分类与模式识别是同义的。如果按产品的若干性能指标将其分为一级品、二级品等，则这可以看成纯分类问题。类别划分的另一种作用是，把具有若干相似特征的事物归属于一个类别，再赋予某种标号和名称，其目的不在于同一类型事物中区分出类别，而在于将具相似特征者聚为一类以便于形成概念、进行操作。例如，为了检索和利用，人们通常将内容相近的资料、文档、图书按类区分，分区储放。这一种类别划分纯依其内容相似程度，而前述的模式识别型的分类则取决于其外在特性，例如前文所举的是或不是癌细胞之例（在这种情况下，即使输入特征差异很大还可能归入一类，而输入特征差异不大也可能分为异类）。在科技、工程领域将模式识别（pattern recognition）和分类（classification）都归于这一型问题。而将纯依内容相似程度

进行的类别划分称为聚类（clustering），将一个函数的输入和输出之间的映射称为回归（regression）问题。

无论是人还是智能机器，学习都是一个关键问题，是重中之重。对学习提出的要求是探索一种学习算法，既有很好的推广性能又有很高的效率。在本节和本章乃至全书，将一再讨论这个问题。

#### 4. 语言能力 听和说 读和写 及视觉能力 分辨物体——颜色、形状、边缘 纹理等，判断空间关系——距离、方向、深度等，以及边缘及线条检测等）

语言既是一种重要的信息交换工具又是知识表示和思维与推理的重要手段。而视觉信息又占了外界输入信息量的 80% 以上，所以大脑皮层中，涉及视觉信息处理者占了很大百分比。

#### 5. 行动和动作

行动就是用腿脚将自身从一个地点移动到另一个地点，这涉及避免与各种静止或活动的障碍物相碰撞以及设计一条从起点到终点的路径。动作就是通过臂和手的协作将一物移至另一处或完成某种动作（扳开关、拧螺丝等）。智能机器人应该和人一样做到这一点。

#### 6. 优化

优化就是根据某一准则制订一个目标函数（或称为价值函数），通过改变或选择各种可供选择的方案、参数、结构或计划 使该函数达到极大值 或极小值）即得到最大“收益”（或产生最小“损失”）。优化问题在人的日常生活和社会生活的方方面面（经济、工业技术、政治、军事……）无不经常出现并需要得到尽好的解决。

#### 7. 预测

人在制订自己的行动计划时，无时无刻不用到预测，预测既包含对外界环境未来的估计，也包含对自身任何行为所会产生后果的估计，并且根据这种预测作出选择。在人类的社会生活中，小至一个工作单位大至一国的政府乃至国际组织、金融机构在制订政策和决定行动时都很强地依赖于预测和预报。

#### 8. 计划、判断和决定

人在完成一定任务时总要做计划和做决定，这实际上是根据任务的要求在已有的知识中进行选择和组合。而选择的标准则取决于优化目标是否达到和对计划或决定执行后果的预测，即必须进行判断。

#### 9. 自适应和鲁棒性

人所生活的环境决非一成不变，且有许多不确定因素，因此一个人学得再多也不可能知道自己的未来所将面临的所有课题。这样人必须随着环境的变化不断修正自己的应对策略，根据自身所做行动受到环境的奖或惩来学习新知识。在遇到从未接触的不定因素时能作出尽量好的抉择，即人的智能行为必须有鲁棒性（robust）和自适应性。

#### 10. 博弈与对策

博弈论与对策论的研究对于揭示人的智能行为的特点和如何针对利益不同的对手（利益相反、利益相同或中性）在博局中的行为给出自身相应对策的策略都极富启发性。特别在经济、军事、社会等许多强竞争领域中，更是对智能行为研究的一种好工具。

## 11. 创新和发明

人的创造能力是智能的一个重要特点，即人可以从学得的知识中通过重新组合、选取和随机变化产生出从未学过的新知识——新的原理、新的机器……

## 12. 深层的智能因素

如意识、感情、意志、注意力、直觉、理解等。

在粗浅地介绍了人脑和人的智能行为后，现在讨论本书的主题——人工神经网络。众所周知，工业革命通过用各种动力机器（蒸汽机、汽油和柴油机、电动机……）代替人的体力劳动开创了一个新时代，使得社会的物质财富有了极大的增长，同时使人们摆脱了过于繁重的体力劳动。现在我们面临着信息时代，这个时代将通过用各种智能机器替代人的智力劳动，来创造一个精神和物质财富远比工业革命时代丰富且使人们过于繁重的智力劳动能够得到减轻的新时代。这个新时代的标志是计算机、包括互联网和移动通信在内的各种通信工具、信号处理（包括视频、音频、文字等）自动控制和智能机器人等许多领域中取得的飞速进展。大规模集成电路的高速发展以及光盘存储容量的持续提高为这个时代提供了强有力的硬件支持，各式各样的信息如潮水般涌来。现在对于能够帮助我们处理各种信息的智能机器的需求十分迫切，例如互联网的快速发展及其多媒体业务和信息量的快速扩展对于网络的智能管理、智能接口（例如语音识别、图像和文字识别等）以及数据挖掘（data mining, DM）的要求越来越高。再如大规模集成电路等现代高精尖产品的质量提高依赖于智能性质量控制、故障检测等。但是，现在所能提供软件的智能水平仍然非常低下。为了应对迫在眉睫的各种智能信息处理课题，20 世纪后半叶开展了多方面的研究与探索，人工神经网络就是在 20 世纪 80 年代初期涌现出来的。经过 20 年来的发展，其成就已蔚为可观。人工神经网络是从生理、心理和认知等各种不同视角出发，对人脑的结构和运行方式进行各种层次的借鉴，以便在计算机上实现具有人类智能行为特点的各层次功能。

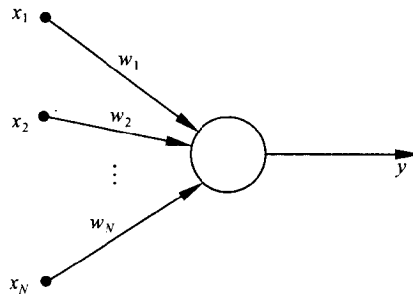


图 1-2 人工神经元

人工神经网络的研究从一开始就借助于神经元的生理结构模型。对于图 1-1 所示神经元的最简单而直接的模仿是图 1-2 所示的人工神经元， $x_1 \sim x_N$  表示来自其他神经元的输入信号， $w_1 \sim w_N$  表示突触强度， $y$  表示本神经元的输出信号，上述各变量和参数皆取实数值。一种简单的输入-输出映射关系是

$$y = \text{sgn} \left[ \sum_{i=1}^N w_i x_i \right] \quad (1-1)$$

其中  $\text{sgn}[\cdot]$  表示取符号值。这就是 20 世纪 40 年代提出的 McCulloch-Pitts 模型<sup>[1]</sup>。下文



将介绍多种其他模型。基于一种特定的神经元模型，可以将很多个神经元按照不同方案组合成各种类型的人工神经网络。各种类型的网络具有如下的一些共同特点。

(1) 网络由同一种简单的非线性处理单元即神经元构成。

(2) 网络的构成和运行都按照分布-并行的方式。每一个神经元和每一个突触强度都承担着所有记忆内容的存储，而每一个记忆项的提取也涉及每个神经元和突触。这就是说人工神经网络的信息存储是分布的，根本找不到某特定信息的存储区；其运行是并行而非串行的，即每一时刻所有神经元同时运行而不是按时序依次进行运算。

(3) 网络中神经元的数量非常巨大，从而使得网络成为一个复杂系统。

(4) 网络有学习能力。它并非按已经编制好的程序来实现某种所需的功能，而是通过学习来获得此功能。

概言之，可以将人工神经网络描述为复杂的、分布-并行的、有学习能力的非线性动力系统。复杂性的概念在当代信息科学、生物学、经济学、物理学和宇宙科学中起重要作用<sup>[2]</sup>。人工神经网络的分布-并行原理使其能用一般速度的处理单元完成极高速的运算，其学习能力和非线性可使之完成过去传统方法无法完成的若干任务。

在智能科学的研究中，与人工神经网络的研究并行且早开始十余年的一项研究是人工智能 AI(artificial intelligence)。AI 采用符号逻辑来构成推理和判断，它利用符号间的语义关系在归纳专家知识的基础上构成“IF...THEN”式的规则库。在解决一个具体智能问题时，采用各种启发式的搜索方案，从知识库（规则库）中寻求解答。其研究重点是如何表示知识，如何进行符号处理，如何进行搜索，如何搜集和归纳专家的知识等。归根到底，其思路是用精确的计算机程序将人类专家的知识归纳和表现出来，计算机用严格安排好的程序按推理方式解决问题，所以人们常称之为专家系统。30多年来，AI研究的主要结论之一是：智能需要知识！这无疑十分正确，但是 AI 在解决实际世界的各种问题时只取得了有限的成功，例如在医疗诊断、弈棋等方面。人工神经网络 ANN(artificial neural network)的研究方法及思路与 AI 形成了鲜明对照，特别在于它借鉴人脑解决智能问题时的生理和心理特点以及通过学习来解决问题。

近十年来，包括 ANN 在内的一个新学科出现并显示出强劲的生命力，这就是计算智能 CI(computational intelligence) 或称为软计算 soft computation)。它包含以下三个相互独立却又相互紧密结合与渗透的子学科：

(1) ANN；

(2) 模糊推理系统 FIS(fuzzy inference system)，或称为模糊集或模糊逻辑；

(3) 进化算法 EA(evolutionary algorithm)，其中包含遗传算法 GA(genetic algorithm)、进化计算 EC(evolutionary computation) 和进化程序 EP(evolutionary program) 等大致相同而略有差异的算法。

其中 FIS 采用模糊集，给予语言一种近似人类思维的数学表示并用模糊逻辑进行推理。FIS 在表示人类知识和认知方面有很强的能力，但是缺乏学习能力。反之，ANN 有强学习能力却很难纳入人类知识。因此，二者结合所构成的模糊神经网络 FNN(fuzzy neural network) 具有明显的优势。EA 是一种并行随机优化算法，它是通过借鉴达尔文的进化论学说构思出来的。在解决许多过去经典优化算法难以解决的高难度、复杂优化问题时，EA 已表

现出很多独特的优势。而 ANN 本身最关键的问题，即学习问题，就是一个高难度、复杂的优化问题。将 EA 用于 ANN 的学习 显然是一个十分理想的方案。不但如此 过去 ANN 的学习主要是针对参数的学习，而结构是通过尝试来决定的。采用 EA 则不但能进行参数学习而且可以进行结构学习，甚至可以进行学习规则的学习。后者在某些方面已接近于人类思维中的创造性范畴了。当今 ANN 的研究已与 FIS 和 EA 的研究分不开，这三项研究的学术会议常在 CI 的大题目下联合举行。文献 [3] 对于该领域的最新进展有 11 篇综述论文予以详细介绍。本书第 5 章专门讨论 FNN 第 6 章讨论 GA 及其在 ANN 学习中的应用。

## 1.2 ANN 的主要模型与研究途径

在 ANN 的研究中提出了很多模型，其差异主要表现在以下几方面。

### 1. 所取的研究途径

指借鉴人脑的哪一种生理或心理的运行特点以及其他相关学科（如物理学、信号处理、统计学等）的何种涉及记忆、学习、分类等的算法。

### 2. 网络中神经元取何种函数

例如，除上文中提过的  $\text{sgn}[\cdot]$  函数外还可以取 Sigmoid 函数、径向基函数 (RBF) 以及更复杂的脉冲耦合神经元模型（见 1.5 节）等。

### 3. 网络的结构

这是指网络中各神经元是如何连接的。在全连接时，每一个神经元与网络中其他神经元之间都有连接。在部分连接时，一个神经元的输入信号只取自网络中一部分神经元。在全连接时，必然存在反馈，即信号存在一个闭合的传输路径。在部分连接时，则可能存在反馈，也可能不存在。

### 4. 运行方式

这涉及很多方面，例如运行的时间变量可取连续值或离散值。如果要更精确地模仿人脑，则需要采取十分复杂的运行方式（见 1.5 节）。

### 5. 学习算法

学习算法很多，概括地可分为有监督的 (supervised) 和无监督的 (unsupervised) 两种，后者又称为自组织的 (self-organized)。这两种算法的差异在于，前者在学习时所赋予的学习样本既含所需完成输入 - 输出映射的输入，又含输出，而后者只含输入不含输出（输出由学习者通过学习予以确定）。

### 6. 应用

可以将 ANN 的应用粗划分为函数逼近（回归）、分类与模式识别、聚类、优化等，还可以更具体地进行划分其实际应用。

ANN 的主要模型介绍如下。

#### 1. 多层前向神经网络 MLFN (multilayer feedforward neural network)

这是一种直至目前研究得最多且应用最广的 ANN，它采用多层局部连接结构、无反馈，神经元函数通常取 Sigmoid 函数或 RBF，一般按离散时间运行 采用有监督学习算法。其应用涵盖面很宽：函数逼近、分类与模式识别、系统辨识与控制、后验概率估计、主分量

分析 PCA(principle component analysis)……它也是本书研究的重点之一。

## 2. 递归神经网络 RNN(recurrent neural network)

这是一种局部连接或全连接、有反馈的网络，其学习、运行、神经元函数选择等方面与 MLFN 类似。主要用途是非线性动力系统的辨识、建模和控制。

## 3. 自组织神经网络 ( self-organized neural network)

这种网络的主要特点是学习算法为无监督的自组织算法，其主要功能是实现对输入特征向量的聚类且在此基础上用于完成函数逼近、分类及模式识别等映射。它的用途的一个明显例证是用发现知识的数据采掘。最著名的两种自组织神经网络是自组织映射 SOM(self-organized mapping ) 网络 and 自适应谐振理论 ART(adaptive-resonance theory) 网络。前者又称 SOFM(self-organized feature mapping) 是芬兰科学家 T. Kohonen 在 20 世纪 80 年代初提出的，目前这种网络在 DM、模式识别和信号处理等领域颇受重视。后者是美国科学家 S. Grossberg 等在 20 世纪 80 年代中期提出的，其特点在于汲取了人脑智能活动的许多心理特点，诸如集中注意、短期与长期记忆、记忆的弹性与刚性、学习与外界奖惩的关系等。

## 4. Hopfield 神经网络

也可简记为 HNN 这种网络是美国物理学家 J. J. Hopfield 在 20 世纪 80 年代初提出的。这是一种全连接反馈网络，其神经元函数为  $\text{sgn}[\cdot]$ ，其运行可按连续时间也可按离散时间进行。前者称为连续时间 HNN，主要用于解决各种优化问题。后者称为离散时间 HNN，主要用于联想记忆、信号的增强与恢复。Liapunov 能量函数的概念在这类网络的研究中起重要作用。

## 5. 模糊神经网络 FNN

FNN 是 ANN 与 FIS 的结合。一种方案是在 MLFN 或 RNN 中纳入 FIS 以使得人的知识能够以 FIS 的形式用于 ANN 的结构设计和参数的粗调整，而 ANN 的学习算法则用于参数细调整。这样就使结构设计和参数学习的效率有很大提高。其应用与 MLFN 及 RNN 相似，特别适用于非线性动力系统的辨识和控制。另一种方案是 FIS 与 SOM 相结合构成模糊聚类系统，与确定聚类系统相比，这种系统对聚类的形成更符合人的认知行为，因此在 DM 这一类应用中性能明显优越。

## 6. 脉冲耦合神经网络 PCNN(pulse coupled NN)

这种网络取更加精确的、更符合实测的生物脑神经元生理-电活动规律的人工神经元模型，从而在神经生理学智能模仿方面向前跨了一大步（见 1.5 节）。目前它主要用于视觉信号的初级处理，如图像分割等，已颇见成效。

## 7. 波尔兹曼机

这是一种全连接、反馈式 ANN，除了其运行规律是随机式的以外，这种网络的结构及用途都与 HNN 类似。其特点是神经元的输出以概率方式依赖于输入且采取模拟退火方案使网络的运行从开始到结束的过程中随机性由大渐次变小，最终变成确定网络。这种网络与人脑的某些活动特点如睡眠、反学习、做梦等有些关连<sup>[4]</sup>

以上介绍的模型是不全面的。例如细胞神经网络 ( cellular NN) 它主要从电路理论的角度出发讨论 ANN 的构成，本书就割爱了。再如 LPN(learning petri network)<sup>[5]</sup> 它

借鉴最近的脑科学知识以及大脑具有不同功能区的特点来构成 ANN。由于这一研究刚起步，故本书也不做讨论。这样的情况很多，就不一一列举了。

## 1.3 ANN 的学习

学习问题对于 ANN 是一个大题目。学习目的和学习算法各式各样，为了节省篇幅，这里只讨论完成输入特征向量至输出变量之间映射的学习。这实际上已涵括了人脑智能系统和在实际应用中要求 ANN 所完成任务的相当大一部分，诸如函数逼近、分类和模式识别以及聚类等。有关学习的问题正如有关知识的问题一样，方面很多，层次很多，不可能尽述。这里只择要予以介绍。

### 1.3.1 三类学习

前文已指出学习可粗分为有监督和无监督两大类。对于前者，在学习开始前，要提供若干对已知输入向量和相应输出变量构成的样本集（或称为训练集），可以认为此集由教师提供，所以称为有教师指导下的学习。其用途为函数逼近、分类和模式识别等。对于后者，学习前只提供若干已知输入向量构成的训练集。其用途为聚类。如做细分，有监督学习还可以分成普通有监督学习和增强学习（reinforcement learning）两类。二者的区别在于对前者的输入-输出映射要求在训练集中规定得十分明确，而对于后者输入-输出映射的输出部分很难规定得十分明确，而只能根据总的待实现目标判断输出的某种变化具有正面效果还是负面效果，而不能给定明确输出值。例如弈棋者以当前棋局为输入向量，下一着棋的弈法为输出，待实现目标为赢棋，即属此种情况。

### 1.3.2 Hebb 学习律

D. O. Hebb 根据生物神经元的工作特点提出了他的著名学习律<sup>[6]</sup>。这可以描述如下。设网络中有编号为  $i$  和  $j$  的两个神经元，它们的输出分别记为  $x_i$  和  $x_j$ ，它们之间的突触强度（连接权）记为  $w_{ij}$ 。再设有编号为  $m = 1, \dots, M$  的  $M$  个训练样本，对于每个样本神经元  $i, j$  的相应输出记为  $x_i^{(m)}, x_j^{(m)}$ ，那么  $w_{ij}$  可用下式计算：

$$w_{ij} = \frac{1}{M} \sum_{m=1}^M x_i^{(m)} x_j^{(m)} \quad (1-2)$$

如果  $x_i$  或  $x_j$  只能取值 1（兴奋）或 -1（抑制）那么可以看到如果在训练中大部分  $x_i^{(m)}$  和  $x_j^{(m)}$  取相同符号，则  $w_{ij}$  取较正之值 反之则取较负之值。各  $w_{ij}$  还可取一种自适应的方式予以修正。设按离散时间  $t$  赋予网络训练样本，在时刻  $t$  神经元  $i, j$  输出为  $x_i(t), x_j(t)$  那么  $t+1$  时刻的  $w_{ij}$  可由  $t$  时刻求得如下：

$$w_{ij}(t+1) = w_{ij}(t) + \alpha x_i(t) x_j(t) \quad (1-3)$$

其中  $0 < \alpha \leq 1$ 。与 Hebb 学习律对立的还有反 Hebb 学习律（anti-Hebb learning）。它是在 (1-3) 式中取  $\alpha < 0$  或者其和式前加负号。反 Hebb 学习往往起遗忘或打乱作用，有些学习算法中借助其跳出常规的途径。

### 1.3.3 目标函数与最陡下降算法

学习问题是一个优化问题。在 ANN 的结构已取定的情况下，学习问题归结为求网络中连接各神经元的权  $w_{ij}$ ，使得一个目标函数达到极小值。此目标函数以某种准则衡量 ANN 对训练集中各输入向量的实际输出与理想输出（在训练集中给定）之间的差异（有监督学习情况），或者训练集中各输入向量的聚类误差（自组织学习情况）。目标函数在统计学中称为经验风险函数（empirical risk function）如果 ANN 中所含的各个权用一个向量  $\xi$  表示，则此函数可表示为  $R_{\text{emp}}(\xi)$  一种求最优  $\xi$  使此函数达到极小的算法是，从一个随机初值  $\xi(0)$  出发按节拍  $k$  进行下列迭代计算：

$$\xi(k+1) = \xi(k) - \alpha \nabla_{\xi} R_{\text{emp}}(\xi) \mid_{\xi=\xi(k)} \quad k = 0, 1, 2, \dots \quad (1-4)$$

这就是最陡下降算法，其中  $0 < \alpha \ll 1$  称为步幅。当  $\alpha$  足够小时，每迭代一步  $R_{\text{emp}}$  将下降或不变当  $k$  足够大时  $R_{\text{emp}}$  将收敛到一个极小值点。

对于 MLFN 而言，用最陡下降算法求最优  $\xi$  的尝试在早期研究中受到了挫折。这是因为无法计算  $R_{\text{emp}}$  相对于各隐含层神经元输入权相应的偏微分，因而不能求得迭代计算中的梯度函数。20 世纪 80 年代前期提出的 BP(back propagation) 算法成功地解决了此问题。这是 ANN 研究蓬勃展开的重要动因之一。最陡下降算法及基于它建立的 BP 算法的优点是非常简单，缺点是因  $\alpha$  必须非常小（否则将导致振荡而不收敛）而使学习速度极慢，另一个缺点是当目标函数存在多个局部极小值点时，如初值选择不当，会收敛到低质局部极小点。即便如此，BP 算法仍是目前使用最广的学习算法之一。

### 1.3.4 学习算法的性能

一种学习算法的性能优劣是指由之产生的 ANN 推广 (generalization) 性能的优劣。这里是指用训练集内数据所确定的 ANN 用于训练集外数据时，其误差若略有增加而差异不大，则推广性能优越；反之，若增加很多则推广性能劣。推广性能至关重要，若 ANN 的推广性能低劣，则其实用价值很低。一个 ANN 推广性能的优劣取决于以下因素。

(1) 待完成的映射任务的复杂度。这主要取决于输入特征向量的维数，维数越高越复杂。也取决于映射函数本身的复杂性。作粗略估计时，就以输入向量的维数估计复杂度。

(2) 训练集的规模。用集中样本的个数来衡量其规模的大小。

(3) ANN 的结构和规模。以 ANN 中所含神经元数以及连接权个数的多寡来衡量。

(4) 学习算法本身。

推广性能与这些因素的关系是，映射任务越复杂或训练集规模越小则推广性能越差；反之则佳。ANN 的规模对于一定的复杂度和训练集规模而言，有一最佳适中值，过大过小都会削弱推广性能。学习算法本身的作用自然是不言而喻的。

### 1.3.5 学习算法的时间效率与 ANN 结构的时空效率

一个学习算法的时间效率是指完成参数学习所耗费的计算机时的多寡。如果所耗机时随着输入特征向量维数的增加而按多项式关系增加，即称为高效率的；反之，如按指数

关系增加,则称低效率的。ANN 结构的效率是指其中所用的神经元个数多少。如果神经元数随输入特征向量维数的增加呈多项式增加,即为高效率;反之,按指数增加时即为低效率。对于 MLFN 常用的两种神经元函数——Sigmoid 函数和 RBF,其空间和时间效率呈完全相反特性<sup>[7]</sup>。当采用 Sigmoid 函数时,每个神经元对其输入向量空间用超平面作全局式的划分,其空间效率高而时间效率低,可以证明这种情况的学习是 NP 完备的复杂问题<sup>[8]</sup>,即学习时间随输入向量维数增加而指数增加。当采用 RBF 时,每个神经元对其输入向量空间用超球作局部式的划分,其空间效率低,即所需神经元数随输入特征向量维数增加而指数增加。在统计学领域中这称为“维数的诅咒”(the curse of dimension)。而其时间效率高,使它十分适合于实时应用的场合。用 Sigmoid 函数的 MLFN 可以逃脱维数的诅咒,但付出了过长学习时间的高昂代价。

### 1.3.6 定向学习和随机学习

按式 1-4) 进行学习计算时,在迭代的每一步目标函数都是非增的。这样,目标函数总是下降并收敛到与初值最接近的参数空间中的一个局部极小点。这称为定向学习。随机学习是在参数随机变化过程中求最优解,这种方法可以避免陷入某个低质局部极小的困境,但是纯随机的搜索显然会耗费过多的时间。更好的方案是定向和随机相结合,在学习时随机成分大,而随着学习的进行随机成分渐减直至只含定向成分。这就是模拟退火 SA(simulated annealing) 的思路。

### 1.3.7 EA 与 ANN 的学习

前文已述及,EA 作为一种并行随机优化算法有很多优点。目前,将 EA 不但用于参数学习而且用于结构学习和学习规则的学习已成为一个重要方向<sup>[9]</sup>。这种方法的研究尚有很大潜力,且处于极初始阶段,特别是结构学习和学习规则的学习更涉及智能深层次方面,值得重视。

### 1.3.8 统计学习理论与支持向量机

ANN 的研究者早已熟知除了网络结构特点、神经元函数的选择和学习算法这些因素外,一个 ANN 推广性能的优劣取决于待完成映射的复杂度、训练集的规模和 ANN 的规模这三个关键因素,特别是在前两个因素一定的条件下,ANN 规模过小或过大都会使推广性能变坏(见 1.3.4 节)。但是,如何在理论上推导和计算出推广性能与此三者的关系却是十分困难的。这个问题由前苏联学者 V. N. Vapnik 用统计学的方法给予了解决,这显示了统计学与 ANN 的结合可以产生极丰富的成果。根据 Vapnik 的统计学习理论(statistical learning theory)<sup>[10]</sup>,包括 ANN 在内的任何学习机的推广误差由理想逼近误差和置信限两部分构成。前者是指某一规模的 ANN 对某一特定复杂度的问题实现映射时,在选择最佳参数的条件下,包括训练集内外所有数据的映射误差平均值。后者是指与训练集内数据不一致的集外数据所引起的额外映射误差平均值。统计学习理论的贡献在于对于某种结构的 ANN 或其他学习机,其规模与一个称为 VCdim 的参数  $h$  相联系,规

模越大  $h$  值越大。如果所用训练集的规模（即集内所含样本数）为  $M$ ，那么可以证明该 ANN 或学习机的置信限正变于  $h/M$ 。如能求得  $h$  就可以用公式将置信限严格计算出来。可以看到  $M$  越大或  $h$  越小则置信限值越低。另一方面，可以严格导出理想逼近误差反变于 ANN 或其他学习机的规模并取决于待实现映射的复杂度。这样，对于任何 ANN 或学习机，只要能根据其规模求得  $h$  在训练集的规模  $M$  和映射复杂度一定的条件下，即可求得置信限、理想逼近误差和推广误差这三者随  $h$  的变化如图 1-3 所示。其中  $h_0$  是使推广误差达到最小的 VCdim，由此可以确定最佳的网络规模。这一理论的难点之一是对不同网络结构的规模计算  $h$ 。

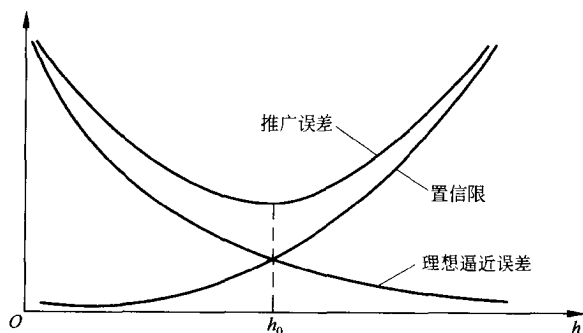


图 1-3 置信限、误差与  $h$  的关系

统计学习理论的另一项重要贡献是支持向量机 SVM(support vector machine) 其用途是困难的模式识别问题，例如人脸识别。SVM 的思路是在输入特征向量空间中寻找划分不同类别的分割面时，分割面两侧分布着训练集不同类别的样本，设与分割面最接近的训练样本与分割面之间的距离是  $\Delta$ ，那么此系统的 VCdim 参数  $h$  与  $\Delta$  成反比。而系统的推广性能又反变于  $h$ 。少数与分割面最接近的那些训练样本的输入特征向量即称为支持向量(support vector) 只有它们能决定  $h$ ，即决定推广性能。问题即可归结为求最佳的分割面及相应的支持向量，使  $\Delta$  达到最大，从而使推广性能达到最佳。SVM 在解决若干实际的困难模式识别问题时取得了令人瞩目的成效，所以备受重视。文献[11]含有多篇论文 讨论了统计学习理论和 SVM 的理论基础、应用和最新进展。

## 1.4 ANN 的应用

ANN 是一门工程科学 在研究 ANN 的进程中必须强调各种实际应用问题的解决。从学科领域讲 ANN 可用于函数逼近、分类和模式识别、聚类、优化、联想和概率密度估计等领域。而本小节主要介绍其实际应用，下面分项陈述。

### 1. 模式识别

这是一个应用极广、门类众多的领域，诸如语音、图像、文字、人脸等的识别对于实现人机智能接口和交互作用是关键的一步。单独或组合使用 MLFN、FNN、SVM、自组织神经网络以及与其他统计方法（如 HMM 等）相结合是提高模式识别系统识别率和鲁棒性的一条有效途径。本书有关各章都举了不少应用实例和参考文献，此处不再详述。

## 2. 非线性动力系统的辨识、建模和控制

传统的经典控制理论对于解决线性动力系统的辨识与控制已提供了较完满的方法，但是涉及非线性和时变系统则问题很多。ANN 的非线性特性和学习能力使其成为解决这方面问题的一个有力候选者。十余年来将 MLFN、RNN、FNN 等用于这一领域的研究已开展很多，文献[12] 对此给出了一个较全面的综述和评价。

与这个课题有联系的是智能控制和智能机器人的问题。这里智能的主要含意是指系统的控制和机器人的行为能通过学习来积累和改善而且随着环境的变化有自适应能力。这正是 ANN 具有优势的方面，见文献[3] 和[13] 中的有关文章。

另一个有关课题是大规模复杂生产流程的质量控制。在化工、大规模集成电路以及其他制造业的产品生产过程中无不包含很多中间环节。每个环节的若干受监视指标与该环节的生产调控参数之间以及它们与最终产品的质量之间，都存在着复杂的非线性映射关系。靠人工来实现非常大量调控参数的调节是很难的，现在已越来越多地使用了 ANN。当然，对于故障诊断和事故报警等需要人智能的工作都能通过 ANN 的使用而提高效率。

## 3. 预测和预报

在一个高度复杂的现代社会中，对未来作较准确的预测与预报对于政府政策的决定和计划的编制都至关重要，尤其是金融、工业生产、消费乃至气候、灾难等的预报和预测。这些领域都涉及千百万人的活动，构成一个非常复杂的非线性系统。近年来将 ANN 用于股市预测和银行坏账预报等已做了不少工作，可以参见本书有关章节提供的文献。

## 4. 数据采掘 DM

DM 是我们这个信息时代以及互联网高速发展时代的一项典型的新技术——从浩如烟海的数据中发掘出具有很高价值的信息和知识。而这一项技术的典型性还在于它是 ANN 模糊集 FS)、统计学和包括数据库技术在内的计算机科学相互结合的产物。由于本书其他部分未设专门章节来讨论它，此处作较详细一些的介绍。

大量数据的出现是信息革命持续高涨的重要标志。据称人类诞生以来总共产生的信息量大约为  $10^{18}$  b 而 1999 年所产生的信息量即占其中的 12% 且其 2/3 为数字化信息即数据(截至 2000 年初的估计)。这些数据产自全球的工业生产、金融、商业、科学和医学等各个最活跃的领域。例如，在能源工业中存储了大量人工地震数据磁带，是否能从中找到开采石油的有用信息呢？在金融和商业领域，市场和消费者的数据每日确如潮水般涌来。在医学界，诊断和治疗的案例可以通过互联网从全世界千百家医院获得。从网上获取科研论文的数量与日俱增。美国 NASA 的地球遥感系统每小时就产生 50Gb 数据(1999 年资料)。而互联网的迅速扩展本身就带来了一种强烈的需求——帮助用户发现所需信息的智能体(intelligent agent)。要从如此大量的数据中找到尽可能多的信息已经超出了人脑的应付能力，这一势头仍在继续，无怪乎有人惊叹这是一个数据丰富而知识贫乏的时代。显然用 CI 的 ANN、FIS 和 EA 来解决此问题是一个合乎逻辑的选择。

DM 可以定义为从数据库 DB(DataBase)中搜索和发现稳定的、有意义的、易于解释的模式，所发现的信息应是新颖的、有应用价值的、有趣的。DM 是 KDD(knowledge discovery in DataBase)的一个核心部分。在 KDD 中实施 DM 以前还要进行数据的选择、预处理并进行适当变换实施 DM 后还要作出解释并给予用户。所需发现或采掘的模式可分为 [15]。



- (1) 聚类。将各种感兴趣的项目 (item) 按相似度划分为不同类别。
- (2) 分类。将各项目按人为的划分归于不同的类别, 其分类标准是外加的。
- (3) 函数关系。项目与数值间建立联系或映射。这也称为回归 (regression)。

(4) 关联 (association)。在项目之间建立因果式的联系, 如  $A$  和  $B$  是由感兴趣的项目建立的两个非交子集 则  $A \rightarrow B$  表示由  $A$  可推出  $B$ 。

DM 最有用的工具是模糊聚类算法, 其中广泛采用的是自组织 ANN-SOM 以及各种模糊推理算法。DM 还用到范畴定向 (context-oriented) 模糊聚类的概念<sup>[15]</sup>

DM 所面临的最困难课题仍是复杂性和推广特性的问题。首先要解决的是用什么特征向量来表示一个文本。文献[16] 给出用一个文本中每个词出现频次直方图来构成 DM 的输入特征向量。可以看到 此特征向量的维数是非常高的 (数千) 为了提高效率必须采取某种变换 (例如基于特征向量分析的单值分解 SVD) 来将此向量的维数压缩下来。另外一方面是为了进行 DM 的学习, 必须有非常大的供训练用数据库以改善推广性能, 这带来很多技术困难。当然, 为了使 KDD 有用, 建立良好的用户交互工具 (特别是采掘结果的可视化) 是极其重要的。DM 仍处于蓬勃发展的初期, 文献[14] 是较近的关于 DM 技术、应用和发展的专辑, 可供参考。

## 5. 优化问题

在工程设计、工程系统运行以及科学研究、经济规划等领域中几乎时时处处都离不开优化。即通过方案、结构和参数的选择使得一个工程或经济系统的效益达到极大值。但是很多实际优化问题往往是 NP 完备的, 即随着待解优化问题的规模 (复杂度) 的增加, 求优化的计算量按指数规律增加。这样, 对于比较复杂的问题求全局最优解是非常困难的。因此任何能求得高质量局部最优解的优化算法都很受重视。连续时间 HNN 在被提出时, 就以解决一个标准优化问题, 即旅行商问题 TSP (travelling salesman problem) 而声名大噪。虽然在过去近 20 年中由于用 HNN 解决优化问题时遇到了一些困难而受到质疑和批评, 对其评价也降低了, 但是由于研究者锲而不舍的努力, 所遇到的困难大部分已被克服。不但如此, HNN 还被扩展到解决一大类优化问题, 这就是具有等式和 / 或不等式约束的 0-1 规划问题, 并在理论和实际两方面都取得了成果。用 ANN 解决各种现代通信系统 (互联网、ATM 网……) 中通信路由管理业务的优化问题就是一个富有成果的实例<sup>[17,18]</sup>。读者在本书中还可以找到其他很多应用实例。

## 6. 信号处理和检测

ANN 可用于语音、图像、雷达、水声等信号处理及检测的各个方面, 且在非线性划分和自适应能力等方面优于传统的统计方法。以图像信号处理为例, ANN 可用于视频压缩<sup>[19]</sup> 图像识别、视觉信号处理 (图像分割、边缘和纹理检测等) 还可以用离散时域 HNN 的联想记忆功能实现从噪声或污染中恢复图像。ANN 用于雷达和水声目标的检测也受到重视<sup>[20]</sup>

## 7. 生物学和医学

生物学和医学与 ANN 的关系是双向的, 一方面前者的成果用来开发更有成效的 ANN, 另一方面后者又可用来进行病症诊断、医学图像分析<sup>[21]</sup> 乃至用来实现基因辨识<sup>[22]</sup> 等。

ANN 的应用还很多 不能尽述。文献[13] 是 ANN 工程应用的一本专辑, 可供参考。需要强调的是除了与 FIS、EA 等相结合外, ANN 应该更好地与各种现代统计算法互相参

透和结合，从而收到事半功倍的效果。此二者相辅相成，不能强调其中一种而排斥另一种。

## 1.5 脉冲耦合神经网络和脑成像技术

近年来通过微电极对猫和猴大脑皮层的测量积累了大量实测资料，研究者从动物视觉皮层的活动特点入手进行分析，得到很多有益启示。以此为基础提出了 PCNN(pulse coupled NN)，它采用更精确的神经元模型和神经元之间连接的集群结构来模仿脑皮层中神经元发放尖脉冲(spike)时的同步现象，从而对大脑的视觉预处理神经机制，特别是景像分割(scene segmentation)机制，作出了更好的解释。这又为遇到很多困难的计算机视觉系统的研发提供了一条颇具潜力的新途径。另一方面，近年来各种脑成像技术的空间和时间分辨率有很大提高，从而为宏观上了解大脑的工作原理提供了条件。因此通过人的大脑成像来研究和开发新型神经网络也是备受重视的方向。由于本书其他章节不涉及这两方面，所以在这一小节介绍这些重要的研究新方向。

### 1.5.1 PCNN

人们早就知道无论人还是动物，脑神经元的工作方式是以三种方式发放尖脉冲：单脉冲发放、周期或周期脉冲串(burst)发放和随机发放。采取何种发放则取决于神经元对各输入信号的积累和神经元本身的状态，这称为积累-发放(integration-firing)机制。近十余年来通过对麻醉状态下的猫和清醒状态下猴大脑皮层的微电极测试，对神经元活动规律及神经元之间如何相互耦合有了更深的了解。下面首先介绍大脑皮层进行视觉信号预处理时的一些特点，然后介绍根据这些特点构成的神经元模型。

最重要的是视觉皮层中存在多个由若干神经元构成的高度同步活动的集团(assembly)，一个集团中的各神经元都接收来自视网膜某个区域的神经元所发出的馈送信号(feeding)，因此它成为视觉空间中某个被覆盖区域的接收场。同时，一个集团中每个神经元又接收同集团中其他神经元发出的链接信号(linking)，所以它又称为一个链接场(linking field)。十分重要的是集团内的各个神经元的活动是同步的，这种活动可能是周期式的(频率在 35Hz ~ 90Hz 之间)，也可能是非周期式的。这些活动既取决于来自集团以外由外刺激产生的馈送信号，又取决于集团内各神经元之间的链接强度以及神经元的动作机理。对于不同的集团之间存在以下几种情况。

(1) 如果两个相邻集团收到相同外刺激产生的馈送信号且二者之间有链接存在，那么两个集团的神经活动会进入快速同步。

(2) 比较靠近的几个集团(这称为局部集团)中，存在一个居间的共同神经元能实现这些集团间的反馈抑制(feedback inhibition)。其作用是有助于实现不相关信号的同步、解除同步或抑制。

(3) 如果分布在近端和远端的若干集团具有相同输入连接(common-input connectivity)，会使靠近或分散的各集团间建立零延时相差(zero-delay phase difference)。

人和其他动物的视觉皮层在处理视觉信号时，首先要完成一个视觉预处理任务，即景像分割。大脑中存在着对图像的不同特征敏感的模块，这些特征包括边缘、角点、方向性、

一致性和不一致性、颜色、光流等。当一个图像中包含其中某些特征时，与这些特征相应的模块被激活起来，从而把这些特征检测出来。然后通过把这些特征重新“组装”起来，在大脑中恢复出一定的景像，从而实现景像分割。值得注意的是当一个感知实体（perceptual entity）被背景弄得有点模糊时（例如有一部分被挡住或遮蔽），在大脑中仍能恢复该实体的完整景像。这说明各特征感觉模块之间存在着关联和相互配合，已检出的特征可以帮助未检出特征的恢复，这样即可由含糊或含混的感觉信号恢复并建立惟一的感知实体。这一过程可描述为特征提取 → 特征链接 → 景像分割。这一工作原理对于视觉信号的前期处理非常必要，这可以使不确定性和无数不重要的细节都被去除或削弱，从而使更高层的执行视觉联想记忆部分有效地解决模式识别等任务。

对上述工作原理作出的最合理解释是皮层中的各个神经元集团，即是对各种特征敏感的接收场和链接场。当一个感知目标或实体出现在视野中时，它的各项特征就激活了各相应的神经元集团，而这些集团通过互相的链接进入同步状态。所有同步活动的特征接收场重构了被感知的实体并实现景像与背景的分割。同步活动是这一解释的核心，正是同步活动使各个被检出特征实现重新拼合。

现在讨论如何构成一个神经元模型来实现这种神经元集团的接收场、链接场、尖脉冲发放和同步建立等各种机制。截至目前为止，所提出的一个较完善的模型是 Reinhard Eckhorn 给出的<sup>[23]</sup> 其结构如图 1-4 所示。下面根据文献[23]、[24] 对其工作原理进行介绍。

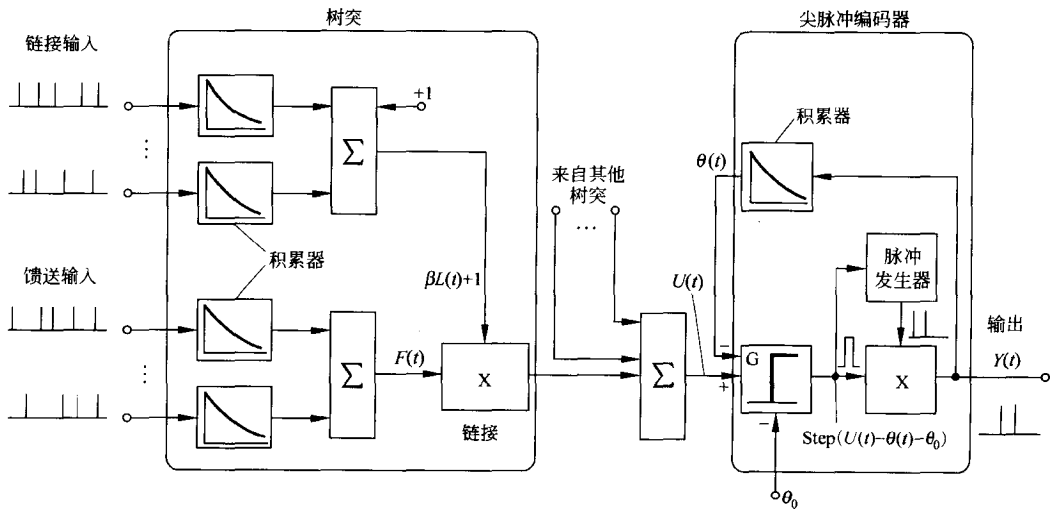


图 1-4 PCNN 神经元模型

PCNN 神经元模型由“链接部分”和“尖脉冲发生器”两部分构成。图 1-4 的左侧是链接部分，它包含若干个树突（图中只示出了一个树突），每个树突又含若干个与视网膜神经元轴突连接的馈送输入突触和若干个与集团内外神经元轴突连接的链接输入突触。图中所示的积累器是具有泄放功能的积分器，它受单位脉冲  $\delta(t)$ <sup>①</sup> 冲击时的响应是  $Ae^{-\alpha t}$  其

此  $\delta(t)$  是 Kronecker  $\delta$  函数，即满足： $\delta(t) = 0$  当  $t \neq 0$ ， $\lim_{t \rightarrow 0} \delta(t) = \infty$ ， $\int_{-\infty}^{\infty} \delta(t) dt = 1$  本章所用  $\delta(t)$  皆具此定义。

中  $A, \alpha$  是两个非负参数。各链接输入信号和各馈送输入信号是一些沿时间轴分布的单位脉冲。如果各馈送信号用  $Y_j(t)$  表示且各链接信号用  $Y_k(t)$  表示(下标  $j$  和  $k$  分别表示视网膜神经元的编号和集团内、外其他神经元编号, 这些馈送和链接信号涉及所有有连接关系的神经元), 那么, 当本神经元的编号为  $i$  时, 它的馈送信号总和  $F_i(t)$  和链接信号总和  $L_i(t)$  可以用下列二式计算<sup>①</sup>

$$F_i(t) = \sum_j M_{ij} Y_j(t) \otimes \Phi_{ij}(t) + I_i \quad (1-5)$$

$$L_i(t) = \sum_k W_{ik} Y_k(t) \otimes \Phi_{ik}(t) + J_i \quad (1-6)$$

其中  $\otimes$  表示卷积。 $\Phi_{ij}(t) = A_{ij} e^{-\alpha_{ij}t}$  和  $\Phi_{ik}(t) = A_{ik} e^{-\alpha_{ik}t}$  分别表示  $ij$  突触和  $ik$  突触之间的积累器的单位脉冲  $\delta(t)$  冲激响应。 $M_{ij}$  和  $W_{ik}$  是突触  $ij$  和  $ik$  之间的突触强度。 $I_i$  和  $J_i$  是两个常值输入。 $\sum$  号下的  $j$  和  $k$  表示对所有与神经元  $i$  有连接的各  $j$  值及  $k$  值取和。链接部分的总输出  $U_i(t)$  可用下式计算:

$$U_i(t) = (1 + \beta_i L_i(t)) F_i(t) \quad (1-7)$$

$\beta_i$  表示链接强度。可以看到, 各馈送信号之间或各链接信号之间是相加的; 而总馈送信号与总链接信号是相乘的。当链接信号不存在时, 可得

$$U_i(t) = F_i(t)$$

尖脉冲编码器部分接收  $U_i(t)$  产生输出  $Y_i(t)$ 。 $Y_i(t)$  也是在时间轴上分布的单位脉冲, 其产生机制介绍如下。首先将  $U_i(t)$ 、可变阈值  $\theta_i(t)$  和固定阈值  $\theta_0$  送入一个阶跃信号发生器, 其输出是  $\text{Step}(U_i(t) - \theta_i(t) - \theta_0)$ 。其中  $\text{Step}(\cdot)$  函数定义为

$$\text{Step}(u) = \begin{cases} 1, & u \geq 0 \\ 0, & u < 0 \end{cases} \quad (1-8)$$

$\theta_i(t)$  是尖脉冲编码器内部产生的,  $\theta_0$  是由其他神经元送来的抑制信号。 $\text{Step}(\cdot)$  函数推动一个脉冲发生器产生周期单位脉冲序列  $\sum \delta(t + nT)$  其中  $T$  取决于神经元从第一次发放后能够进行再次发放的间歇期, 是一固定值;  $n$  的取值范围是  $-\infty \sim +\infty$  之间的整数。 $\text{Step}(\cdot)$  与此周期序列相乘即得到神经元  $i$  的输出  $Y_i(t)$ ,

$$Y_i(t) = \text{Step}(U_i(t) - \theta_i(t) - \theta_0) \left( \sum \delta(t + nT) \right) \quad (1-9)$$

$Y_i(t)$  送到一个积累器, 产生可变阈值  $\theta_i(t)$ 。可以用下列微分方程来描述其运行过程:

$$\frac{d\theta_i(t)}{dt} = -\alpha_T \theta_i(t) + V_T Y_i(t) \quad (1-10)$$

其中  $\alpha_T$  和  $V_T$  是两个正参数。易于求得此积累器受单位脉冲  $\delta(t)$  冲激时的响应是  $V_T e^{-\alpha_T t}$ 。

现在用此神经元模型来解释在 PCNN 中一个孤立神经元是如何产生孤立的单尖脉冲和尖脉冲串的(在下面的讨论中都假设  $\theta_0 = 0$ )。先讨论前者, 其运行原理如图 1-5 所示。设  $L_i(t) = 0, F_i(t)$  处于一个低值, 这时  $U_i(t) = F_i(t)$  亦为低值。设  $\theta_i(t) > F_i(t)$  这时  $\text{Step}(U_i(t) - \theta_i(t)) = 0$  所以  $Y_i(t) = 0$ 。随着  $\theta_i(t)$  的逐渐衰减, 在其达到略小于  $F_i(t)$  的

① 此处只考虑一个树突的情况, 本节下文中的讨论与此相同。

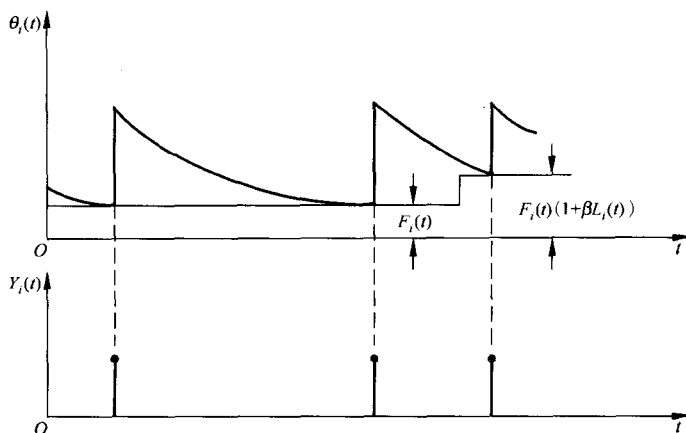


图 1-5 神经元的工作原理图

一刻,  $\text{Step}(U_i(t) - \theta_i(t)) = 1$ , 从而产生一个尖脉冲。此脉冲使  $\theta_i(t)$  迅速提高并重新使  $\text{Step}(U_i(t) - \theta_i(t)) = 0$  这再次使神经元停止发放 即  $Y_i(t) = 0$ 。然后 随着  $\theta_i(t)$  的缓慢衰减, 在过了很长间隔后又产生一个尖脉冲。如果在其后链接信号的来到使  $L_i(t) \neq 0$  这时  $F_i(t)(1 + \beta L_i(t))$  较  $F_i(t)$  高, 从而使下一次尖脉冲的发放提前了。

如果一个神经元处于一个集团之中, 那么由于集团中每个神经元都收到集团内其他神经元送来的链接信号, 这时就会产生密集同步脉冲串发放, 其产生机制如图 1-6 所示。

可以看到 当  $\theta_i(t)$  衰减至小于  $F_i(t)$  时 不但神经元  $i$  产生发放而且集团内其他神经元都产生发放, 从而产生很大的链接信号  $L_i(t)$ 。这样就使  $U_i(t)$  在发放时升高的值远超

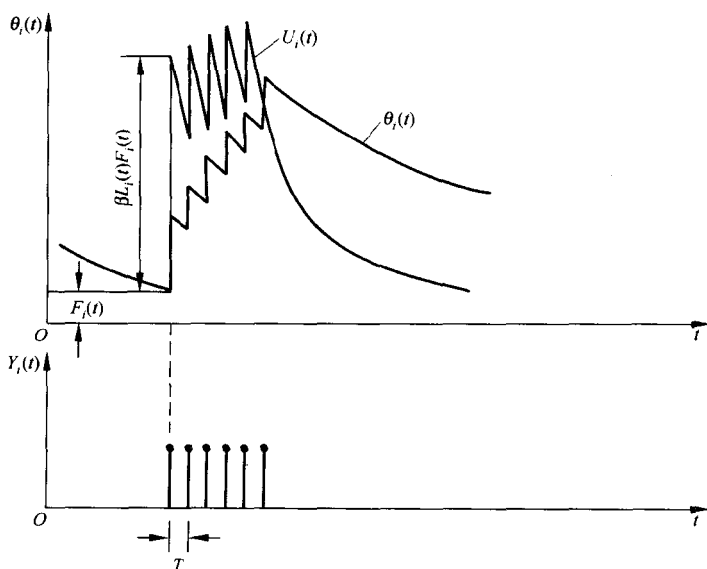


图 1-6 密集同步脉冲串发放机制

过  $\theta_i(t)$  所升高之值。由于  $\text{Step}(U_i(t) - \theta_i(t))$  保持为 1 神经元就以  $T$  为间隔持续发放。随着发放的进行  $\theta_i(t)$  呈阶梯式上升直至某一时刻  $\theta_i(t)$  超过了  $U_i(t)$ 。这时集团内所有神经元都沉寂下来。在此以前,集团内的神经元都同步地发放了一个密集的脉冲串。这里关于 PCNN 的讨论是初步的,各个集团之间如何相互链接或抑制的机制相当复杂,此处不再涉及。应该说 PCNN 的研究尚属起步阶段 文献[25]是关于 PCNN 的一个专辑,可以参考。

### 1.5.2 大脑成像与 ANN

相对于以微电极测试为基础而建立的 PCNN,ANN 基于各种脑成像技术,可以对人的大脑作全局图像式的描绘,而前者只能对猫或猴的大脑局部功能作出阐释。研究者认为这正在引发对大脑运行机制了解的一场革命。其源头是各种高分辨率脑成像机器和各种数据分析技术的出现,这使得人们对全局大脑有了崭新的看法。从人脑图像可以确定,大脑由特定激活的神经模块(module)组合而成,而这些模块又是各种模块以不同方式、针对不同应用耦合而成的,各模块执行着相当恒定的任务,有的对输入感觉信号作前期处理,有的执行更高层的任务,诸如集中注意、语言、思考、意识等。对研究者而言是找到在大脑巨大而复杂的功能背后隐藏的运行程序。对于 ANN 研究者,这种研究提供了建立新的网络模型的依据。另一方面 ANN 也可以检验一些脑功能运行假说是否合理并且提供对大脑图像数据分析的有效技术(例如盲信号处理中的独立分量分析 ICA)。功能脑图像研究的另一个重要意义在于,它所取得的是处于清醒状态下的人脑图像,因而能对人的感知、认识和运动动作等机理和人的认知功能有更透彻的了解。

现在所用的脑成像技术分成以下两类。

#### (1) 基于血流动力学 —— 新陈代谢法

其中包括正电子辐射断层成像 PET(positron emission tomography) 单光子辐射断层成像 SPET(single photon emission tomography) 以及功能磁共振成像 FMRI(functional magnetic resonance imaging)。

#### (2) 基于电磁原理

其中包括脑电图 EEG 和脑磁图 EMG。

这两类技术的特点是第一类的空间分辨率高而时间分辨率低,而第二类与之相反。当然,还需采用先进的数据处理技术来提高分辨率和去除各种加性和卷积性质的干扰。

这项研究正处于蓬勃发展阶段,文献[26]、[27]给出了全面的综述。特别是文献[28]为关于脑成像技术与 ANN 建模的一本专辑,可供参考。

## 1.6 ANN 与盲信号处理

盲信号处理 BSP(blind signal processing)是 ANN 与统计信号处理以及信息论相结合的产物,它是 20 世纪最后十年迅速发展起来的一个新研究领域。作为上述几门学科相互渗透和相互融合而产生的新学科,它极具典型意义 —— 在不同学科的边缘结合处极有可能产生新学术观点、方法,从而开辟一条崭新的学术研究途径。BSP 的背景是,在现实世界中所得信号往往是不“纯”的信号。早期信号处理研究中所涉及的不纯信号只是在原始

纯信号上迭加如高斯噪声这种简单情况，并针对这种情况采取各种线性滤波算法尽量去除加性噪声以恢复原始的纯信号。但是实际世界的不纯信号较此要复杂得多。一种情况是接收的不纯信号由多个原始纯信号各乘以相应加权系数后迭加而成（一般还要迭加高斯噪声，可以把它也算做一个原始纯信号）。在实际应用中一般可以用多个接收器来接收多个不纯信号，每个不纯信号是多个原始纯信号的某种线性加权组合。现在提出的问题是：在这些组合关系未知的条件下，是否能找到一种算法，可以恢复各个原始纯信号。这就是盲信号分离 BSS(blind signal separation) 它是 BSP 的重要组成部分。很明显 BSS 非常重要。在生物学和医学的脑电图 EEG、心电图 ECG 及上节所述的其他大脑成像技术中都遇到此类问题，在雷达和水声的相控阵雷达信号接收系统以及地球物理信号处理（探油）等领域也都涉及相同的问题。另一种情况是，接收的不纯信号由一个纯原始信号与一个未知的信道响应卷积而成，这时恢复原始信号的任务称为盲解卷（blind deconvolution）。更复杂的情况是，多个纯原始信号与各自未知的信道响应卷积后再加权组合构成接收信号时（一般是多个接收信号），各个原始信号的恢复问题。这称为多道盲解卷。盲解卷和多道盲解卷也是 BSP 的重要组成部分。除了在医学、地球物理等领域外，它在通信、图像和语音信号处理、财经、数据分析和压缩等许多领域都极具应用价值。

在 ANN 研究界 从上世纪 80 年代中就开始有人将 MLFN 用于信号分离的研究。这就是主分量分析 PCA (principle component analysis)。但是 PCA 只用到信号的二阶统计特性，所以分离效果不佳。而统计信号处理学界所提出的独立分量分析 ICA(independent component analysis)，却能够充分利用信号的高阶统计特性，但它缺乏有效的学习算法。上世纪 90 年代初，这两个领域的研究相互结合后产生了 BSP，它的优越性能与应用潜力使之成为 ANN 发展的一个新热点，十年中取得了长足的进展。迄今为止，对 BSP 的研究和应用仍处于发展阶段，有许多问题尚未解决。本书第 7 章专门讨论 BSP 在该章可以找到有关的文献，这里就不再列出了。

## 1.7 本书的组织

除了本章绪论外，本书其他 6 章的内容简介如下。第 2 章讨论 MLFN。如前所述，MLFN 是研究得比较透彻，算法比较完备，同时也是应用最广的一种 ANN。本书将 RNN 也划归这一章。对于各种模式识别、非线性动力系统辨识与控制、函数逼近、预测等应用感兴趣的读者可以重点读这一章。对于研究而言，推广特性和统计学习理论（包括 SVM）的研究尚有很大潜力。由多网络法构成 MLFN 是一个值得重视的研究方向<sup>[7]</sup>。此外，学习算法的研究仍远未结束，如最近提出将统计学的 EM 算法用于 MLFN 的研究刚刚开始<sup>[7]</sup>。第 3 章研究两种主要的自组织神经网络——SOM 和 ART，它们的主要功能是聚类。其中 SOM 由于算法简单、有效，十几年来一直受到高度重视、广泛研究和应用。在信号处理和模式识别等领域，它往往用来对输入信号进行预处理，即通过聚类有效地压缩输入向量的维数，这对于后续 ANN 的性能（识别率和推广特性）和空、时效率的提高至关重要。聚类与近年来人们对大脑皮层的前期信号处理神经模块（集团）的研究一样关键。最近备受重视的数据挖掘也以 SOM 的聚类算法或者 SOM 与 FS 相结合而构成的模糊聚类算法作为

核心技术。ART 由于其早期算法过于复杂而难以在实际应用中推广。本书只介绍其改进的简化算法，而更详细的介绍可以在文献[4]中找到。第4章介绍 HNN。连续时间 HNN 在 20 世纪 80 年代由于在解决优化问题时的优越表现而备受重视。此后一段时间，它由于在解决实际优化问题时遇到了若干障碍而受到质疑。但是，并非如某些人所预料，HNN 将一蹶不振了。相反，它的大部分障碍业已克服而且应用领域还在推广。目前它是一种有竞争力的优化算法，特别在复杂的通信系统优化和管理中得到应用。离散时间 HNN 的联想记忆功能，无论对于人脑机制的深层了解，还是对信号处理中被污染信号的恢复，以及对有效信号提取等各种理论和实际问题都很有价值。从算法看，近年来进展也很大。总之，HNN 绝不是明日黄花，而仍是一个值得重视的研究课题。第5章介绍模糊神经网络 FNN。如前文所指 FNN 是各种类型 ANN 与各种模糊算法相结合的产物。本书主要讨论其中两种。一种是 MLFN 或 RNN 与 FIS 相结合形成的 FNN。目前已提出了几种有效的此种 FNN 学习算法，其基本思路是用 FIS 进行网络结构设计和参数粗调，而用 ANN 学习算法进行参数细调。大量研究表明，这种 FNN 的性能和学习效率均较单纯的 ANN 或 FIS 有大幅改善。其应用主要是非线性动力系统辨识与控制、函数逼近和模式识别等。另一种是 SOM 与 FS 结合构成模糊聚类算法，已在前文述及。第6章讨论遗传算法 GA 及其在 ANN 参数、结构以及学习算法学习中的应用。有关 EA 或 GA 的书已很多，该章除了给出 GA 的一些基本知识外，重点放在 GA 与 ANN 的结合上。这个课题的提出已经有不短时间，但是待研究的问题仍很多，人们对此领域的兴趣有增无减。第7章讨论盲信号处理 BSP。该章除了给出 BSP 的背景材料外对 BSP 各个领域的理论和算法作了详尽介绍。既可供实际使用者直接应用，也可供有兴趣做进一步研究者参考。

本书的各章内容都是独自完整可读的，无论从学习、研究和应用的哪个角度出发，都可选择其中一章或几章来阅读。我们建议读者研究各种 ANN 时要重视思路和应用以及文后所附的参考文献。在学习的基础上还应该查阅更多的最新文献，因为就在本书编写至出版的一两年间各个领域和课题又有了很多进展。由于 ANN 的涉及面很宽，作为研究而言最好只专注于一个方向、一种模型、一种算法或一项应用，待有了结果再推广至其他。此处我们再次强调在 ANN 的学习与研究中应更好地与统计学、统计信号处理和信息论等相关学科相结合，这可以使工作更有成效。



## 参 考 文 献

- [1] McCulloch W S, Pitts W. A logical calculus of the ideas immanent in neurous activity. Bulletin of Mathematical Biophysics, 1943, 5: 115 ~ 133
- [2] Mitchell Waldrop 著 陈玲译. 复杂诞生于秩序与混沌边缘的科学. 北京: 三联书店, 1997
- [3] Special issue on computational intelligence. Proceedings of the IEEE, Sep. 1999, 87 (9)
- [4] 杨行峻, 郑君里. 人工神经网络. 北京: 高等教育出版社, 1992
- [5] Hirasawa Kataro, et al. Learning Petri network and its application to nonlinear system control. IEEE Trans. on SMAC—Part B: Cybernetics, Dec. 1998, 28 (6): 781 ~ 789
- [6] Hebb D O. The organization of behavior. New York: Wiley, 1949
- [7] Ma S, Ji C. Performance and efficiency: recent advances in supervised learning. Proceedings of the IEEE, Sep. 1999, 87(9): 1519 ~ 1535
- [8] Blum A L, Rivest R L. Training a 3-node neural network is NP-complete. Neural Networks, 1992, Vol. 5: 117 ~ 127
- [9] Yao X. Evolving artificial neural networks. Proceedings of the IEEE, Sep. , 1999, 87(9): 1423 ~ 1447
- [10] Vapnik Vladimir N. An overview of statistical learning theory. IEEE Trans. on Neural Networks, Sep. 1999, 10(5): 988 ~ 999
- [11] Special issue on Vapnik-Chervonenkis (VC) learning theory and its applications. IEEE Transactions on Neural Networks, Sep. 1999, Vol. 10, No. 5
- [12] Narandra K S. Neural networks for control: theory and practice. Proceedings of the IEEE, Oct. 1996, 84(10): 1385 ~ 1406
- [13] Gelenbe E, Barhen J. ed. Special issue on engineering applications of artificial neural networks. Proceedings of the IEEE, Oct. 1996, 84(10)
- [14] Special issue on neural networks for data mining and knowledge discovery. IEEE Trans. on Neural Networks, May 2000, 11(3)
- [15] Hirota K, Pedrycz W. Fuzzy computing for data mining. Proceedings of the IEEE, Sep. 1999, 87(9): 1575 ~ 1599
- [16] Kohonen T, et al. Self organization of a massive document collection. IEEE Trans. on Neural Networks, May 2000, 11(3): 574 ~ 585
- [17] Habib I W. Applications of neurocomputing in traffic management of ATM networks. Proceedings of the IEEE, Oct. 1996, 84(10): 1430 ~ 1441
- [18] Ambrose B E, Goodman R M. Neural networks applied to traffic management in telephone networks. Proceedings IEEE, Oct. 1996, 84(10): 1421 ~ 1429
- [19] Cramer C, et al. Low bit-rate video compression with neural networks and temporal subsampling. Proceedings IEEE, Oct. 1996, 84(10): 1529 ~ 1543
- [20] Sham S. Neural network optimization for multi-target multi-sensor passive tracking. Proceedings IEEE, Oct. 1996, 84(10): 1442 ~ 1457
- [21] Glenbe E, et al. Neural network methods for volumetric magnetic resonance imaging of the human brain. Proceedings of the IEEE, Oct. 1996, 84(10): 1488 ~ 1496
- [22] Xu Y, et al. GRAIL: a multi-agent neural networks system for gene identification. Proceedings of

the IEEE, Oct. 1996, 84(10); 1544 ~ 1552

- [23] Eckhorn Reinhard. Neural mechanism of scene segmentation; recordings from the visual cortex suggest basic circuits for linking field models. IEEE Trans. on Neural Networks, May 1999, 10(3): 464 ~ 479
- [24] Johnson L, Padgett M L. PCNN models and applications. IEEE Trans. on Neural Networks, May, 1999, 10(3); 480 ~ 498
- [25] Special issue on PCNN. IEEE Trans. on Neural Networks, May, 1999, 10(3)
- [26] Taylor J G. Towards the networks of the brain; from imaging to consciousness. Neural Networks, 1999, 12; 943 ~ 959
- [27] Horowitz B, et al. Neural modelling and functional brain imaging; an overview. Neural Networks, 2000, 13; 829 ~ 846
- [28] Special issue on the global brain; imaging and modelling. Neural Networks, 2000, 13(10); 827 ~ 1061

# 第 2 章 前向多层神经网络与递归神经网络

## 2.1 概 述

前向多层神经网络 (multilayer feedforward neural network, MLFN) 是目前应用最广、发展最迅速的人工神经网络之一。MLFN 的研究从 20 世纪 60 年代已经开始,但是由于当时找不到恰当的学习算法,这一研究曾长期处于低潮。直到上世纪的 80 年代中期出现了 BP(back-propagation) 学习算法,有效地解决了 MLFN 的学习问题,从而极大地推动了这一领域的研究工作。近十年来 MLFN 的理论研究和实际应用都达到了很高的水平。

### 2.1.1 MLFN 的结构和运行原理

MLFN 采取一种单向多层结构,其中每一层包含若干个神经元,同一层的神经元之间没有相互联系,层间信息的传送只沿一个方向进行。一般采取由底向上的方式描述这一结构,它包括一个输入层 (层号  $l = 0$ )、包括  $L - 1$  个隐层 (层号  $l = 1 \sim L - 1$ ) 以及一个输出层 (层号  $l = L$ )。第  $l$  层包含  $N_l$  个神经元,每个神经元的输出记为  $x_i^{(l)}, i = 1 \sim N_l$  从;它们构成一个行向量  $\mathbf{X}^{(l)} = [x_1^{(l)}, \dots, x_{N_l}^{(l)}]$ 。在计算网络层数时,输入层一般不计算在内。输入向量  $\mathbf{X}^{(0)} = [x_1^{(0)}, \dots, x_{N_0}^{(0)}]$ 。一般用  $\mathbf{X} = [x_1, \dots, x_N]$  表示,其中  $N = N_0$ ,  $\mathbf{X} \in \mathbb{R}^N$ 。输出向量  $\mathbf{X}^{(L)} = [x_1^{(L)}, \dots, x_{N_L}^{(L)}]$  一般用  $\mathbf{Y} = [y_1, \dots, y_P]$  表示,其中  $P = N_L$ ,  $\mathbf{Y} \in \mathbb{R}^P$ 。对于每一个输入向量  $\mathbf{X}^{(0)} = \mathbf{X}$ ,由底向上逐层求得  $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}$  直至求得  $\mathbf{X}^{(L)} = \mathbf{Y}$ ,这一过程称为前向计算。参见图 2-1 所示的网络结构示意图。

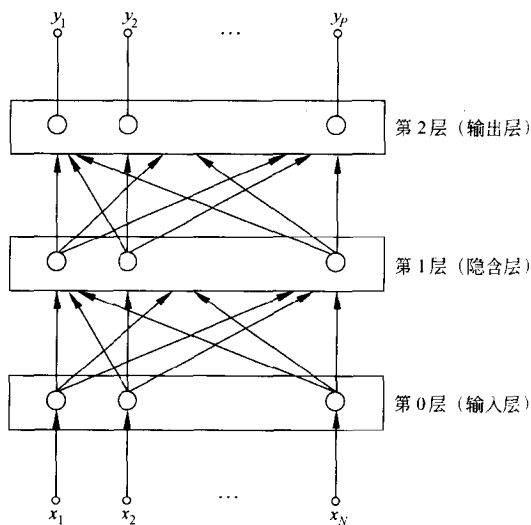


图 2-1 MLFN 的网络结构图

网络的层数  $L$  取决于很多因素,通常  $L$  取 2 或 3;对于一些复杂的问题,  $L$  可以取 4~6。若  $N = N_0 > 1, P = N_L > 1$  网络称为 MIMO- $L$  层 MLFN 若  $N = N_0 > 1, P = N_L = 1$ , 网络称为 MISO- $L$  层 MLFN。

### 2.1.2 MLFN的功能

概括而言,一个 MLFN 的功能是实现  $X$  至  $Y$  的映射,可以表示为  $X \rightarrow Y$ 。作更细的划分,可以分成以下几类功能。

#### 1. 模式识别

若只需将  $X$  分成两种类别,可以用 MISO 网络。当输出  $y_1 = 1$  表明  $X$  属于某一类;当  $y_1 = 0$  表明  $X$  属于另一类。若需将  $X$  分成  $P$  种类别,则可以用  $N_L = P$  的 MIMO 网络。网络的  $P$  个输出  $y_i = x_i^{(L)}, i = 1 \sim P$  中只有一个允许为 1 其他必须为 0 当  $y_l = 1$  且  $y_i = 0, i \neq l$ , 表明  $X$  属于第  $l$  类。

#### 2. 函数逼近

若  $X \in \mathbb{R}^N, Y \in \mathbb{R}^P, P > 1$ , 网络实现的映射关系可以表示为  $Y = F(X)$  若  $P = 1$ , 则  $Y = y$  此关系为  $y = f(X)$ 。为简化起见下面只讨论后者。设待实现的映射关系是  $y = f(X)$  则希望  $f(X)$  与  $f(X)$  之间的差异尽可能小。

#### 3. 条件概率估计

设有随机向量  $X \in \mathbb{R}^N$  和随机事件  $C_i, i = 1 \sim P$ 。二者的关系可以用其联合概率分布  $p(C_i, X)$  描述,它可以用条件概率分布函数来表示为(贝叶斯公式)

$$p(C_i, X) = p(X/C_i)P(C_i) = P(C_i/X)p(X)$$

实际中常需对  $p(X/C_i)$  或  $P(C_i/X)$  作出估计。若用  $X$  作为一个具有  $P$  个输出的 MLFN 的输入向量,则这  $P$  个输出值可以逼近于  $P(C_i/X)$ 。若  $P(C_i)$  和  $p(X)$  已知,则可用上述贝叶斯公式由  $P(C_i/X)$  求得  $p(X/C_i)$ 。

#### 4. 两个离散时间序列之间的映射

设有两个离散时间序列  $X(n)$  和  $Y(n), n = 0, 1, 2, \dots, X(n)$  是神经网络的输入  $Y(n)$  是其输出。可以用映射  $X(n) \rightarrow Y(n), n = 0, 1, 2, \dots$  来逼近任何需要实现的映射  $X(n) \rightarrow \hat{Y}(n), n = 0, 1, 2, \dots$ 。如果对于任一时刻  $n, \hat{Y}(n)$  不仅依赖于  $\dots, X(n-1), X(n)$  而且依赖于  $\dots, Y(n-2), Y(n-1)$  这时必须在一般 MLFN 中增加一些延时反馈环节,才能更好地逼近这种序列之间的映射。这种网络称为递归神经网络(recurrent neural network, RNN)。如果  $Y(n)$  与  $n$  时刻以前的  $\hat{Y}(n-2), \hat{Y}(n-1)$  等无关,则可用 2 所述的无反馈映射完成,即每个时刻  $n$  的  $X(n)$  映射为对应的  $Y(n)$  这是一种“静态映射”。而 RNN 完成“动态映射”。

### 2.1.3 MLFN 的应用

MLFN 的应用几乎涉及科学、工业技术、金融、医学、农业、气象、地矿等各个领域。例如,在字符、图像、语音、癌细胞、心脑电信号、伪钞等许多识别和分类任务中大量应用了 MLFN。再如,当代的各种制造行业和工业生产工程(例如石化工程)日趋复杂化,对产品的质量、制造精度和成品率的要求日益提高,采用计算机进行设计、过程监控、质量控制乃至故障诊断已成为必然趋向,这就需要通过实现智能控制、机器人操作和计算机集成制造系统

(CIMS) 等新技术,MLFN 在其中的作用日益扩大。MLFN 可用于非线性动力系统的辨识、调节和控制,用于参数提取和故障诊断等。以集成电路 (IC) 芯片的制造为例,一个芯片的制造涉及几百个环节,任何一步不慎都会使产品报废,为了对几千个生产过程参数进行严格控制以保证质量,必须采用 IC-CIMS。例如其中的等离子光刻环节是将芯片的多晶硅膜置于一个气室中,在高频电波 (RF) 和作用化学气体 ( $\text{CCl}_4$ ) 的联合作用下进行刻蚀。此环节的控制参数是:RF 功率、气室压力、气体组成、电极间隔等,而效果参数是:刻蚀速度、刻蚀均匀度、选择性等。在理论上很难建立一个模型由前者求后者,而 MLFN 可以在二者之间建立映射关系,效果优于其他方法。类似的要求包括实时闭环控制和故障诊断等。在化工、能源、航空甚至彩色胶卷生产等领域中都存在类似问题。在信息和信号处理领域中,MLFN 用于信道均衡、信号检测、编码、信息安全、系统智能控制和管理等许多方面。在经济和金融领域中,可用于坏账判断、股市预测等许多方面。

### 2.1.4 MLFN 中的神经元模型

MLFN 中任一神经元的模型如图 2-2 所示 其输出  $x_i^{(l)}$  用下式来计算:

$$\left. \begin{aligned} x_i^{(l)} &= f_l[I_i^{(l)}] \\ I_i^{(l)} &= \varphi_l(\mathbf{X}^{(l-1)}, \mathbf{W}_i^{(l)}, \theta_i^{(l)}) \end{aligned} \right\} \quad (2-1)$$

其中  $f_l[\cdot]$  是  $l$  层的神经元函数,  $I_i^{(l)}$  是  $l$  层第  $i$  神经元的净输入,  $\mathbf{W}_i^{(l)}$  和  $\theta_i^{(l)}$  是其权向量和阈值参数,  $\mathbf{W}_i^{(l)} = [w_{i1}^{(l)}, \dots, w_{iN_{l-1}}^{(l)}]$ , 各  $w_{ij}^{(l)}$  和  $\theta_i^{(l)}$  皆取实数值,  $\mathbf{X}^{(l-1)}$  是第  $l-1$  层的输出向量。

常用的神经元函数有下列四种。

#### 1. 线性函数

$$f_l[I_i^{(l)}] = I_i^{(l)} \quad (2-2)$$

$$\varphi_l(\mathbf{X}^{(l-1)}, \mathbf{W}_i^{(l)}, \theta_i^{(l)}) = \mathbf{X}^{(l-1)} \cdot \mathbf{W}_i^{(l)} + \theta_i^{(l)} \quad (2-3)$$

$\mathbf{X} \cdot \mathbf{Y}$  表示两个向量的点积 (内积)。

#### 2. 指示函数 (硬限幅函数)

$$f_l[I_i^{(l)}] = \begin{cases} 1, & I_i^{(l)} \geq 0 \\ 0, & I_i^{(l)} < 0 \end{cases} \quad (2-4)$$

$r_l(\cdot)$  仍按式 (2-3) 计算。

#### 3. Sigmoid 函数 (S 形函数)

$\varphi_l(\cdot)$  按式 (2-3) 计算,  $f_l[\cdot]$  有下列两种形式

(1)

$$f_l[I_i^{(l)}] = [1 + \exp(-I_i^{(l)})]^{-1} \quad (2-5)$$

其最大值为 1 最小值为 0 且  $f_l[0] = \frac{1}{2}$ 。

(2)

$$f_l[I_i^{(l)}] = \tanh[I_i^{(l)}] \quad (2-6)$$

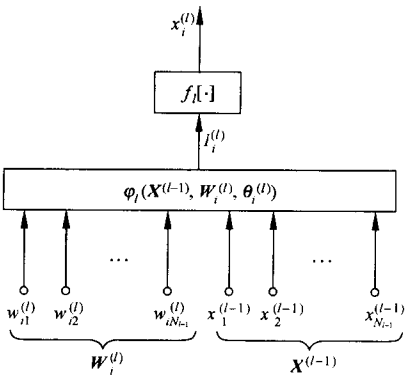


图 2-2 MLFN 中的神经元模型

其最大值为 1 最小值为 -1 且  $f_i[0] = 0$ 。

#### 4. 径向基函数 RBF(radial basis function)

常用的 RBF 有下列两种形式。

(1)

$$f_i[I_i^{(l)}] = (\sqrt{2\pi} \sigma_i^{(l)})^{-1} \exp(-I_i^{(l)}), \quad \sigma_i^{(l)} > 0 \quad (2-7)$$

$$I_i^{(l)} = \varphi_i(\mathbf{X}^{(l-1)}, \mathbf{W}_i^{(l)}) = \frac{\|\mathbf{X}^{(l-1)} - \mathbf{W}_i^{(l)}\|^2}{2(\sigma_i^{(l)})^2} \quad (2-8)$$

(2)

$$f_i[I_i^{(l)}] = (2\pi |\boldsymbol{\Sigma}_i^{(l)}|)^{-\frac{1}{2}} \exp(-I_i^{(l)}) \quad (2-9)$$

$$I_i^{(l)} = \varphi_i(\mathbf{X}^{(l-1)}, \mathbf{W}_i^{(l)}) = \frac{1}{2}(\mathbf{X}^{(l-1)} - \mathbf{W}_i^{(l)})(\boldsymbol{\Sigma}_i^{(l)})^{-1}(\mathbf{X}^{(l-1)} - \mathbf{W}_i^{(l)})^T \quad (2-10)$$

其中  $\boldsymbol{\Sigma}_i^{(l)}$  是一个  $N_{l-1} \times N_{l-1}$  对称正定矩阵,  $|\boldsymbol{\Sigma}_i^{(l)}|$  是其行列式值,  $T$  表示转置。

### 2.1.5 MLFN 的研究和应用中的三大问题

#### 1. 学习问题

以 MISO 网络为例, 设其输入为  $\mathbf{X}$  相应的网络输出为  $y = f(\mathbf{X}, \boldsymbol{\xi})$ ,  $\boldsymbol{\xi}$  表示网络中所有的权和阈值参数构成的行向量; 再设与  $\mathbf{X}$  相应的理想输出为  $\hat{y}$ 。网络采取有监督的方式进行学习: 首先给出  $M$  对学习样本  $(\mathbf{X}_m, \hat{y}_m)$ ,  $m = 1 \sim M$  它们称为一个容量为  $M$  的训练集。其次若网络输入为  $\mathbf{X}_m$  时的实际输出为  $y_m = f(\mathbf{X}_m, \boldsymbol{\xi})$  且  $y_m$  与理想输出  $\hat{y}_m$  之间的损失为  $L(\hat{y}_m, y_m)$ , 那么学习任务是求得最佳参数向量  $\boldsymbol{\xi}_a$  使得下列的经验风险函数  $R_{\text{emp}}(\boldsymbol{\xi})$  达到最小值,

$$R_{\text{emp}}(\boldsymbol{\xi}) = \frac{1}{M} \sum_{m=1}^M L(\hat{y}_m, y_m) = \frac{1}{M} \sum_{m=1}^M L(\hat{y}_m, f(\mathbf{X}_m, \boldsymbol{\xi})) \quad (2-11)$$

一种最常用的损失函数定义是

$$L(\hat{y}_m, y_m) = (\hat{y}_m - y_m)^2 \quad (2-12)$$

学习问题归结为以尽可能小的计算量求得  $\boldsymbol{\xi}_a$ 。

#### 2. 规模问题

仍以 MISO 网络为例 设待实现的由  $\mathbf{X}$  至  $y$  的映射用函数形式表示为  $y = \hat{f}(\mathbf{X})$  而网络实际实现的映射可以表示为  $y = f(\mathbf{X}, \boldsymbol{\xi})$ ,  $\boldsymbol{\xi} \in \Delta$  这里  $\boldsymbol{\xi}$  不仅可以表示一参数行向量, 还可以表示 MLFN 的层数  $L$  和每层的神经元数  $N_l$  等网络结构规模参数  $\Delta$  是其定义域。规模问题是指应如何选择规模参数  $L$  和  $N_l$ ,  $l = 1 \sim L-1$  以使得  $f(\mathbf{X}, \boldsymbol{\xi})$  和  $\hat{f}(\mathbf{X})$  足够接近 这是一个函数逼近问题。一般采取以下两种准则来衡量二者的接近程度。设  $\mathbf{X} \in \mathcal{D} \subset \mathbb{R}^N$  则

(1) 设一个 MLFN 的层数及神经元函数固定不变, 若  $f(\mathbf{X})$  满足某些条件 当隐层神经元足够多时, 对于任何  $\epsilon > 0$  总能使式

$$\|f - \hat{f}\|_u \stackrel{\text{def}}{=} \sup_{\mathbf{X} \in \mathcal{D}} |f(\mathbf{X}, \boldsymbol{\xi}) - \hat{f}(\mathbf{X})| < \epsilon \quad (2-13)$$

成立, 那么该网络具有均匀逼近性质。 $\stackrel{\text{def}}{=}$  表示定义为,  $\sup$  表示取上界。

(2) 在与 (1) 的前提相同条件下, 若隐层神经元足够多, 对任何  $\epsilon > 0$  总能使式

$$\|f - \hat{f}\|_{L_2} \stackrel{\text{def}}{=} \int_{\mathcal{X}} [f(\mathbf{X}, \xi) - \hat{f}(\mathbf{X})]^2 d\mathbf{X} < \epsilon \quad (2-14)$$

成立, 那么该网络具有均方逼近性质。

当神经元取一些常用函数时, 网络的逼近问题已经解决, 这由下列定理陈述。

**Cybenko 定理** (见参考文献 [2])

若 MLFN 的隐层用 Sigmoid 函数且输出层用线性函数, 层数  $L = 2$  单输出。若  $f(\mathbf{X})$  为有限支 (或有限) 单值连续函数, 则网络能对其实现均匀逼近和均方逼近。

**RBF 网络的逼近定理** (见文献 [39]、[40]、[41])

若 MLFN 的隐层用 RBF 且输出层用线性函数, 层数  $L = 2$  单输出。若  $f(\mathbf{X})$  为有限支单值连续函数, 则网络能对其实现均匀逼近和均方逼近。

若 2 层 MLFN 的隐层神经元取小波分析中使用的尺度函数且输出层取线性函数时, 可以逼近任何有限支单值连续或非连续函数 (详见 2.6 节)。

此处未讨论逼近速度的问题, 这涉及  $f(\mathbf{X})$  的性质、 $\epsilon$  和隐层神经元数之间的关系 (详见 2.9 节)。

还应指出, 虽然对许多实际问题 2 层 MLFN 是一种优选方案, 但并不排斥在其他一些情况采用 3 层甚至更多层网络, 可能具有更好的学习性能或推广性能。

### 3. 推广问题

在 1 中网络通过对训练集的学习求得的最佳参数  $\xi_a$ , 可以使训练集内的实际输出与理想输出之间的平均损失, 即经验风险, 达到最小。对于训练集外的数据, 其平均损失是否仍然保持足够小呢? 这就是推广问题。网络推广性能不好就没有实用价值。用有限的训练集数据求一套最佳参数后, 推而广之用于无穷尽的集内外数据, 这属于一般的归纳推理 (inductive inference) 学习课题。这可以说是包括 MLFN 在内的所有学习机的重要的研究课题。按照统计学习理论的观点, 所有输入向量  $\mathbf{X}$  及相应理想输出  $\hat{y}$  的集合可以用其联合概率分布函数  $P(\mathbf{X}, \hat{y})$  描述。设网络对于  $\mathbf{X}$  的实际输出为  $y = f(\mathbf{X}, \xi)$  那么  $y$  与  $\hat{y}$  之间损失对全集的平均值称为风险函数并用  $R(\xi)$  表示:

$$R(\xi) = \int L(\hat{y}, f(\mathbf{X}, \xi)) dP(\mathbf{X}, \hat{y}) \quad (2-15)$$

设最佳参数  $\xi_0$  使  $R(\xi)$  达到最小, 已知  $\xi_a$  使经验风险  $R_{\text{emp}}(\xi)$  达到最小, 如果  $R(\xi_a)$  接近于  $R(\xi_0)$ , 表明由经验风险最小归纳原理求得的最佳参数也能使全局的风险函数接近于最小, 即网络有良好推广性能。在什么条件下  $R(\xi_a)$  能接近于  $R(\xi_0)$  是统计学习理论讨论的内容 (详见 2.10 节)。

学习、规模和推广是互相联系和制约的。例如 网络的规模越大 学习越困难。再如 网络的推广性能与规模密切相关, 在训练集容量一定的条件下, 若其规模超出或小于一个恰当时, 其推广性能明显变坏。

## 2.1.6 本章讨论的内容

MLFN 按照其神经元所取的函数可分成几个子类 取线性函数、硬限幅函数和 Sigmoid 函

数时称 MLP(multilayer perceptron) ;取 RBF和线性函数组合时称 RBF 网络 取小波函数和线性函数组合时称小波网络。本章将讨论这些网络的特点、应用和学习算法。重点将放在 MLFN 推广能力的理论研究和实际提高的方法。此外要介绍一些新研究方向 在 MLFN基础上加延时反馈构成递归神经网络 RNN) 用 MLFN 求解 PCA 问题和 MLFN 的前端信号处理。

## 2.2 线性函数 MLP

### 2.2.1 线性函数单神经元的 LMS 学习算法

设有一单神经元,其输入为行向量  $\mathbf{X}, \mathbf{X} = [x_1, \dots, x_N] \in \mathbb{R}^N$  其输出为  $y \in \mathbb{R}^1$ 。其输入至输出的映射关系为线性函数,即

$$y = \mathbf{W} \cdot \mathbf{X} = \mathbf{W}\mathbf{X}^T \quad (2-16)$$

$\mathbf{W} = [w_1, \dots, w_N] \in \mathbb{R}^N$  是一权向量。若对于  $\mathbf{X}$  的理想输出用  $\hat{y}$  表示 理想输出  $\hat{y}$  与实际输出  $y$  之差用  $\epsilon$  表示 即

$$\epsilon = \hat{y} - y = \hat{y} - \mathbf{W}\mathbf{X}^T \quad (2-17)$$

设  $\mathbf{X}$  和  $\hat{y}$  都是随机变量  $\epsilon^2$  相对于  $\mathbf{X}$  和  $\hat{y}$  的全体集合取平均值是  $\mathbf{W}$  的函数 记为  $R(\mathbf{W})$  即  $R(\mathbf{W}) = E[\epsilon^2], E[\cdot]$  表示对集合取平均,易于证明

$$R(\mathbf{W}) = \beta + \mathbf{W}\mathbf{Q}\mathbf{W}^T - 2\mathbf{P}\mathbf{W}^T \quad (2-18)$$

其中  $\beta = E[\hat{y}^2] \geq 0; \mathbf{Q} = E[\mathbf{X}^T \mathbf{X}]$  是  $\mathbf{X}$  的相关阵,这是一个  $N \times N$  维对称正定阵;  
 $\mathbf{P} = E[\hat{y}\mathbf{X}]$ 。

设有最佳权向量  $\mathbf{W}_0$  使  $R(\mathbf{W})$  达到最小值  $\mathbf{W}_0$  应满足

$$\nabla_{\mathbf{W}} R(\mathbf{W}) \big|_{\mathbf{W}=\mathbf{W}_0} = 2\mathbf{W}_0\mathbf{Q} - 2\mathbf{P} = 0$$

由此可以解得

$$\mathbf{W}_0 = \mathbf{Q}^{-1}\mathbf{P} \quad (2-19)$$

由于  $\mathbf{Q}$  为正定阵  $\mathbf{Q}^{-1}$  存在且是惟一的 所以  $\mathbf{W}_0$  存在且是惟一的。为了避免矩阵求逆的困难 可以从任何初值  $\mathbf{W}(0)$  出发 按照节拍  $k = 0, 1, 2, \dots$  进行下列递推计算来求  $\mathbf{W}_0$ :

$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha E[\epsilon(k)\mathbf{X}(k)] \quad (2-20)$$

可以证明(见参考文献[1])若  $\mathbf{Q}$  的最大特征值为  $\lambda_{\max}$  则  $\alpha$  满足

$$0 < \alpha < 1/\lambda_{\max} \quad (2-21)$$

时,  $\mathbf{W}(k)$  随着  $k$  的增大必然趋向于  $\mathbf{W}_0$ 。为了避免式(2-20)中仍存在的求统计平均的困难,可以采用下列随机逼近递推计算:

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \alpha(k)\epsilon(k)\mathbf{X}(k), k = 0, 1, 2, \dots \quad (2-22)$$

其中  $\epsilon(k) = \hat{y}(k) - y(k), y(k) = \mathbf{W}(k) \cdot \mathbf{X}(k)$ 。可以证明(见文献[1]):只要  $\alpha(k)$  满足下列条件 随着  $k$  的无限增大  $\mathbf{W}(k)$  收敛于  $\mathbf{W}_0$  的概率等于 1。 $\alpha(k)$  称为步幅函数,它满足

$$\sum_{k=0}^{\infty} \alpha(k) = \infty, \quad \sum_{k=0}^{\infty} \alpha^2(k) < \infty \quad (2-23)$$

可以看到  $\alpha(k) = 1/k$  是满足此条件的一种步幅函数。



### 2.2.2 线性函数 MLP(L 层)

在网络中所有神经元皆取线性函数时，网络中第  $l$  层输出向量  $\mathbf{X}^{(l)}$  可以通过第  $l-1$  层输出向量求得；即有

$$\mathbf{X}^{(l)} = \mathbf{X}^{(l-1)} \mathbf{W}^{(l)}$$

其中  $\mathbf{W}^{(l)}$  是一个  $N_{l-1} \times N_l$  维矩阵。这样 输入向量  $\mathbf{X}^{(0)}$  至输出向量  $\mathbf{X}^{(L)}$  的映射关系是

$$\mathbf{X}^{(L)} = \mathbf{X}^{(0)} \mathbf{W}^{(1)} \mathbf{W}^{(2)} \dots \mathbf{W}^{(L)} = \mathbf{X}^{(0)} \mathbf{W}$$

其中  $\mathbf{W}$  是一个  $N_0 \times N_L$  维矩阵。因此 只从输入至输出映射的角度看 无论  $L$  取多大值 此网络相当于  $N_L$  个线性函数单神经元的功能。也就是说，它只能实现线性映射关系。这种网络的主要研究和应用方向之一是用于主分量分析，即 PCA( 详见 2.8.4 小节 )

## 2.3 硬限幅函数 MLP

### 2.3.1 硬限幅函数单神经元的分类功能

设神经元的输入是  $\mathbf{X} = [x_1, \dots, x_N]$  输出是  $y$  权向量是  $\mathbf{W} = [w_1, \dots, w_N]$ 。若式 (2-4) 中的硬限幅函数  $f_l[\cdot]$  用  $\text{sgn}[\cdot]$  表示，那么此神经元的输入至输出映射关系可写成

$$y = \text{sgn}[\mathbf{W} \cdot \mathbf{X} + \theta] \quad (2-24)$$

其中  $\theta$  是阈值。下式定义了  $\mathbb{R}^N$  空间的一个超平面：

$$\mathbf{W} \cdot \mathbf{X} + \theta = 0$$

按照式 (2-24),  $\mathbb{R}^N$  被此超平面分割成两部分。当  $\mathbf{X}$  在超平面某一侧时，输出  $y = 1$ ；在另一侧时， $y = 0$ 。

图 2-3 所示是  $\mathbb{R}^2$  中的分割超平面（一条直线）。

如果在一个模式识别课题中需要将  $\mathbb{R}^N$  中的向量分成两类，当此二类所占空间可以用一超平面分开时，称其为线性可分类情况；反之，则称非线性可

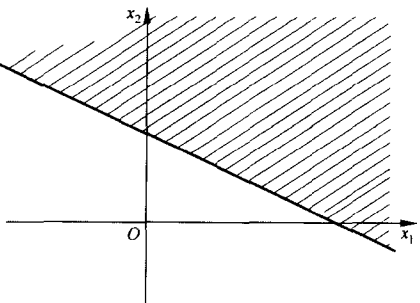
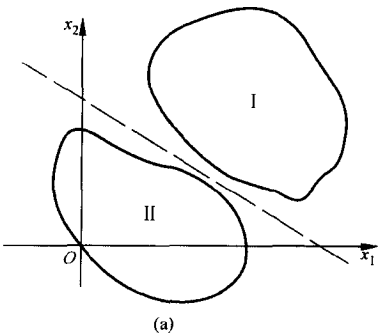
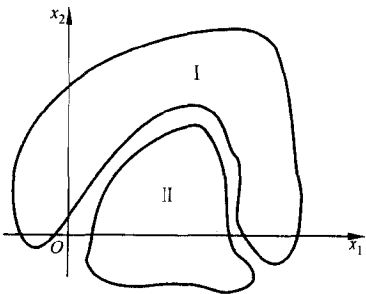


图 2-3 超平面的分割



(a)



(b)

图 2-4 两种情况的示例

(a) 线性可分类情况；(b) 非线性可分类情况

分类情况。图 2-4(a) 和 (b) 分别是这两种情况的示例 ( $\mathbb{R}^2$  情形)。

对于线性可分类情况，可以用  $y$  指示  $\mathbf{X}$  所属的类别且能做到无误划分。

2.3.2 硬限幅函数 2 层单输出 MLP 的分类功能

如果  $\mathbf{X}$  分成两类，分别处于  $\mathbb{R}^N$  中一个凸空间的内部和外部（判断一个空间是否凸空间的方法是在该空间中任择二点，若二点连线上任一点仍属该空间，则此空间为凸；反之为非凸）。硬限幅函数 2 层单输出 MLP 可对这二种类别实现无误划分。图 2-5 所示是  $\mathbb{R}^2$  中的一个凸空间，图 2-7 所示是  $\mathbb{R}^2$  中一个非凸空间。现以图 2-5 为例 证实上述命题。

设图 2-5 中凸空间内外的  $\mathbf{X}$  分别是第 I 和第 II 类。图 2-6 所示是用来进行分类的 MLP。隐层的 3 个神经元与图 2-5 中  $l_a, l_b, l_c$  三条直线对应。当  $\mathbf{X}$  在  $l_a$  的凸空间一侧时，要求  $x_1^{(1)} = 1$  反之  $x_1^{(1)} = 0$ 。当  $\mathbf{X}$  在  $l_b$  的凸空间一侧时，要求  $x_2^{(1)} = 1$  反之  $x_2^{(1)} = 0$ 。当  $\mathbf{X}$  在  $l_c$  的凸空间一侧时，要求  $x_3^{(1)} = 1$  反之  $x_3^{(1)} = 0$ 。隐层至输出的映射为： $y = \text{sgn}\left[\sum_{j=1}^3 w_j^{(2)} x_j^{(1)} + \theta^{(2)}\right]$ 。若置  $w_j^{(2)} = 1, j = 1 \sim 3, \theta^{(2)} = -2.5$ ，那么当  $\mathbf{X}$  位于凸空间中时，因为  $x_j^{(1)} = 1, j = 1 \sim 3$ ，所以  $y = 1$ ；当  $\mathbf{X}$  在凸空间外时，因为各  $x_j^{(1)}$  中至少有一个等于 0，所以  $y = 0$ 。这样  $y$  的值正确指示了  $\mathbf{X}$  所属类别。这一结论可推广到凸空间有任意有限多个超平面边界及  $N$  取任意有限值的情形。

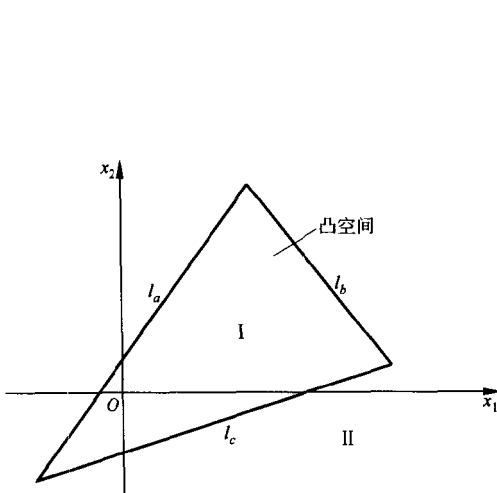


图 2-5  $\mathbb{R}^2$  中的一个凸空间

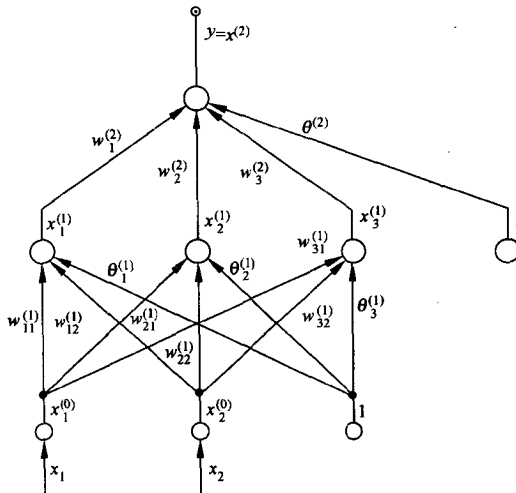


图 2-6 MLP 示意图

2.3.3 硬限幅函数 3 层单输出 MLP 的分类功能

如果两类  $\mathbf{x}$  所占的空间互不重叠，则此网络可予以无误划分。以图 2-7 所示的非凸空间为例，其内外分属两个类别。前述 2 层 MLP 对此不能正确分类。这时可采用图 2-8 所示的 3 层 MLP 第一隐层的 5 个神经元输出  $x_i^{(1)}, i = 1 \sim 5$  与图 2-7 中的直线  $l_a, l_b, l_c, l_d$ ,

$l_e$  对应。第二隐层的输出  $x_1^{(2)} = 1$  时指示  $\mathbf{X}$  属于由  $l_a, l_b, l_c$  围成的凸空间;  $x_1^{(2)} = 0$  则否。 $x_2^{(2)} = 1$  指示  $\mathbf{X}$  属于  $l_c, l_d, l_e$  围成的凸空间;  $x_2^{(2)} = 0$ , 则否。网络输出  $y = \text{sgn}[w_1^{(3)} x_1^{(2)} + w_2^{(3)} x_2^{(2)} + \theta^{(3)}]$ 。置  $w_1^{(3)} = w_2^{(3)} = 1, \theta^{(3)} = -0.5$  当  $\mathbf{X}$  在由  $l_a, l_b, l_d, l_e$  围成的非凸空间内时,  $x_1^{(2)}$  和  $x_2^{(2)}$  中至少有一个等于 1 因此  $y = 1$  当  $\mathbf{X}$  在其外时,  $x_1^{(2)}$  和  $x_2^{(2)}$  皆等于 0 因此  $y = 0$ 。这样, 此网络可实现这一非凸空间内外的正确分类。这一结论可推广到任意两个非交叠空间及  $N$  取任何有限值的分类问题。

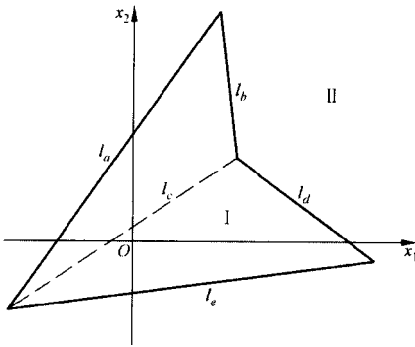


图 2-7 一个非凸空间

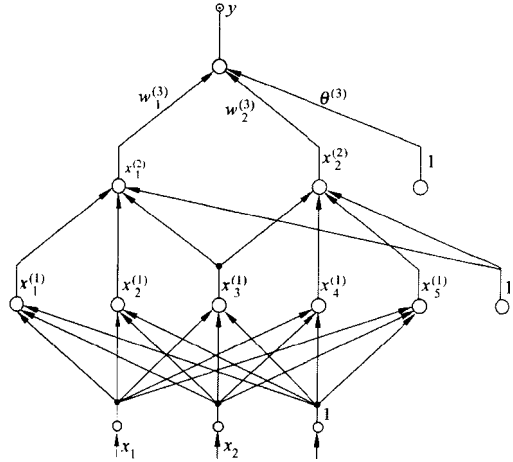


图 2-8 3 层 MLP 示意图

硬限幅函数 MLP 不存在有效的学习算法, 从而限制了它的实际应用范围。

## 2.4 用 Sigmoid 函数和线性函数的 MLP

### 2.4.1 网络结构

本节讨论  $L$  层单输出 MLP 其隐层神经元取 Sigmoid 函数, 输出层神经元取线性函数或 Sigmoid 函数 这两种情况的分析方法区别不大 本节中只分析前者 图 2-9 给出了一个 2 层单输出 MLP 的示例。

对于一个  $L$  层单输出 MLP, 输入向量为  $\mathbf{X} = [x_1, \dots, x_N]$ ,  $\mathbf{X}^{(0)} = [x_1^{(0)}, \dots, x_{N_0}^{(0)}]$ .  $\mathbf{X}^{(0)} = \mathbf{X}, N_0 = N$ . 各隐层输出向量为  $\mathbf{X}^{(l)} = [x_1^{(l)}, \dots, x_{N_l}^{(l)}], l = 1 \sim L-1$ . 输出为  $\mathbf{X}^{(L)} = [x_1^{(L)}], x_1^{(L)} = y$ . 由  $l-1$  层向量  $\mathbf{X}^{(l-1)}$  求  $l$  层向量  $\mathbf{X}^{(l)}$  的计算公式是

$$x_i^{(l)} = f_s[\mathbf{X}^{(l-1)} \cdot \mathbf{W}_i^{(l)} + \theta_i^{(l)}], i = 1 \sim N_l, l = 1 \sim L-1 \quad (2-25)$$

$$x_1^{(L)} = \mathbf{X}^{(L-1)} \cdot \mathbf{W}_1^{(L)} + \theta_1^{(L)} \quad (2-26)$$

其中  $f_s[\cdot]$  是式(2-5) 或式(2-6) 给出的 Sigmoid 函数 (本节只讨论前者),  $\mathbf{W}_i^{(l)} = [w_{i1}^{(l)}, \dots, w_{iN_{l-1}}^{(l)}]$  是各神经元的权向量,  $\theta_i^{(l)}$  是阈值。这样, 当各权向量和阈值给定时, 就能由任何输入向量  $\mathbf{X} = \mathbf{X}^{(0)}$  计算得输出  $y = x_1^{(L)}$  这是前向计算。

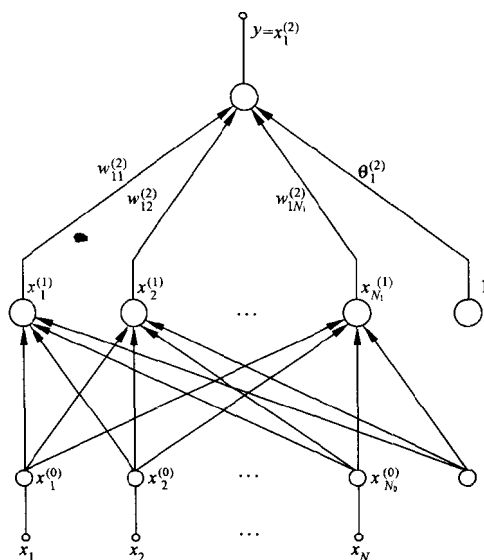


图 2-9 2 层单输出 MLP 的示例

## 2.4.2 网络的 BP 学习算法 (离线批处理)

网络的输入至输出映射： $y = f(\mathbf{X}, \xi)$ ,  $\xi \in \Lambda$ ,  $\Lambda$  是  $\xi$  的定义域。若给定训练集： $(\mathbf{X}_m, \hat{y}_m)$ ,  $m = 1 \sim M$ , 按照式 (2-11) 和式 (2-12) 的定义, 学习目的是求最佳参数  $\xi_a$  使下列风险函数  $R_{\text{emp}}(\xi)$  达到极小值:

$$R_{\text{emp}}(\xi) = \frac{1}{M} \sum_{m=1}^M (\hat{y}_m - f(\mathbf{X}_m, \xi))^2 = \frac{1}{M} \sum_{m=1}^M (\hat{y}_m - x_{m1}^{(L)})^2 \quad (2-27)$$

为此, 采用最陡下降算法。首先设置随机初值  $\xi(0)$  再按节拍  $k = 0, 1, 2, \dots$  进行迭代计算

$$\begin{aligned} \xi(k+1) &= \xi(k) + \Delta \xi(k) \\ \Delta \xi(k) &= -\alpha \nabla_{\xi} R_{\text{emp}}(\xi) \Big|_{\xi=\xi(k)} \end{aligned} \quad (2-28)$$

其中  $\alpha > 0$  称为步幅。若  $\alpha$  取较小值 使得  $\|\Delta \xi(k)\| \ll 1$  ( $\|\cdot\|$  表示一个向量的模值), 则有

$$R_{\text{emp}}(\xi(k+1)) = R_{\text{emp}}(\xi(k)) + \Delta R_{\text{emp}}(\xi(k))$$

$$\text{其中 } \Delta R_{\text{emp}}(\xi(k)) \approx \Delta \xi(k) \cdot \nabla_{\xi} R_{\text{emp}}(\xi) \Big|_{\xi=\xi(k)} = -\alpha \left\| \nabla_{\xi} R_{\text{emp}}(\xi) \Big|_{\xi=\xi(k)} \right\|^2 \leq 0$$

这表明 只要  $\alpha$  足够小  $R_{\text{emp}}(\xi(k))$  将随着  $k$  的增加而下降, 因此  $\xi(k)$  将趋向于  $R_{\text{emp}}(\xi)$  的一个局部极小点。若此点的  $R_{\text{emp}}(\xi)$  值足够小, 即可以把它作为所需解  $\xi_a$ 。

在递推计算中, 式 (2-28) 可表示为各个权和阈值的递推计算 (二者类似 只列前者):

$$w_{ij}^{(D)}(k+1) = w_{ij}^{(D)}(k) + \Delta w_{ij}^{(D)}(k)$$

其中

$$\Delta w_{ij}^{(l)}(k) = -\alpha \left. \frac{\partial R_{\text{emp}}(\xi)}{\partial w_{ij}^{(l)}} \right|_{\xi=\xi(k)} \quad (2-29)$$

这样, 递推计算归结为求上式右侧的偏微分。引用式 (2-27) 得到

$$\left. \frac{\partial R_{\text{emp}}(\xi)}{\partial w_{ij}^{(l)}} \right|_{\xi=\xi(k)} = \frac{-2}{M} \sum_{m=1}^M \left[ (\hat{y}_m - x_{m1}^{(L)}) \frac{\partial x_{m1}^{(L)}}{\partial w_{ij}^{(l)}} \right] \Big|_{\xi=\xi(k)} \quad (2-30)$$

其中  $x_{m1}^{(L)}$  由前向计算求得,  $\hat{y}_m$  在训练集中给定, 现在问题归结为求下列偏微分,

$$\left. \frac{\partial x_{mq}^{(l')}}{\partial w_{ij}^{(l)}} \right|_{\xi=\xi(k)}, \quad q = 1 \sim N_{l'}, l \leq l', l' = 1 \sim L$$

只要能够计算它, 便能计算式 (2-30) 右侧的偏微分。引用式 (2-1) 至 (2-3) 及式 (2-5) 得到

$$x_{mq}^{(l')} = f_{l'}[I_{mq}^{(l')}] = f_{l'} \left[ \sum_{p=1}^{N_{l'-1}} w_{qp}^{(l')} x_{mp}^{(l'-1)} + \theta_q^{(l')} \right] \quad (2-31)$$

$$\frac{\partial x_{mq}^{(l')}}{\partial w_{ij}^{(l)}} = \frac{df_{l'}[I_{mq}^{(l')}]}{dI_{mq}^{(l')}} \frac{\partial I_{mq}^{(l')}}{\partial w_{ij}^{(l)}} \quad (2-32)$$

上式右侧第 2 项可分下列两种情况来计算。

(1)  $l = l'$

$$\frac{\partial I_{mq}^{(l')}}{\partial w_{ij}^{(l)}} = \begin{cases} x_j^{(l'-1)}, & q = i \\ 0, & q \neq i \end{cases} \quad (2-33)$$

(2)  $l < l'$

$$\frac{\partial I_{mq}^{(l')}}{\partial w_{ij}^{(l)}} = \sum_{p=1}^{N_{l'-1}} w_{qp}^{(l')} \frac{\partial x_{mp}^{(l'-1)}}{\partial w_{ij}^{(l)}} \quad (2-34)$$

若  $l' - 1 = l$  则式 (2-34) 右侧各偏微分可以用式 (2-32) 和 2-33 求得。

② 若  $l' - 1 > l$  则式 (2-34) 右侧各偏微分可以用式 (2-32) 和再次使用式 (2-34) 求得。式 (2-32) 右侧第 1 项也可分下列两种情况计算。

(1)  $f_{l'}[\cdot]$  为线性函数 (式 2-2)) 则

$$\frac{df_{l'}[I_{mq}^{(l')}]}{dI_{mq}^{(l')}} = 1 \quad (2-35)$$

(2)  $f_{l'}[\cdot]$  为第一种 Sigmoid 函数 (式 2-5)) 则

$$\frac{df_{l'}[I_{mq}^{(l')}]}{dI_{mq}^{(l')}} = f_{l'}[I_{mq}^{(l')}] (1 - f_{l'}[I_{mq}^{(l')}]) = x_{mq}^{(l')} (1 - x_{mq}^{(l')}) \quad (2-36)$$

全部计算可以规整为下列便于进行编程计算的形式。由式 (2-29) 和式 (2-30) 每一节拍  $k$  的权调整量可用下式计算:

$$\Delta w_{ij}^{(l)}(k) = \frac{2\alpha}{M} \sum_{m=1}^M (\hat{y}_m - x_{m1}^{(L)}(k)) \frac{\partial x_{m1}^{(L)}}{\partial w_{ij}^{(l)}} \Big|_{\xi=\xi(k)} \quad (2-37)$$

令  $x_{m1}^{(L)}$  对  $I_{mi}^{(l)}$  的偏导数为  $\delta_{mi}^{(l)}$  并用式 (2-3) 得到

$$\frac{\partial x_{m1}^{(L)}}{\partial w_{ij}^{(l)}} = \frac{\partial x_{m1}^{(L)}}{\partial I_{mi}^{(l)}} \cdot \frac{\partial I_{mi}^{(l)}}{\partial w_{ij}^{(l)}} = \delta_{mi}^{(l)} x_{mj}^{(l-1)} \quad (2-38)$$

这样, 式 2-37 可改写为

$$\Delta w_{ij}^{(l)}(k) = \frac{2\alpha}{M} \sum_{m=1}^M (\hat{y}_m - x_{m1}^{(l)}(k)) \delta_{mi}^{(l)}(k) x_{mj}^{(l-1)}(k) \quad (2-39)$$

$\delta_{mi}^{(l)}(k)$  分两种情况计算。

(1) 当  $l = L, i$  只能等于 1,  $x_{m1}^{(L)}(k) = f_L[I_{m1}^{(L)}(k)]$ , 利用式 (2-35) 和式 (2-36) 得到

$$\delta_{m1}^{(L)}(k) = \begin{cases} 1, & f_L[\cdot] \text{ 为线性函数} \\ x_{m1}^{(L)}(k)(1 - x_{m1}^{(L)}(k)), & f_L[\cdot] \text{ 为第一种 Sigmoid 函数} \end{cases}$$

(2) 当  $l = 1 \sim L-1$  按以下推导, 由  $\delta_{mp}^{(l+1)}(k), p = 1 \sim N_{l+1}$  求  $\delta_{mi}^{(l)}(k), i = 1 \sim N_l$

$$\begin{aligned} \delta_{mi}^{(l)}(k) &= \frac{\partial x_{m1}^{(l)}(k)}{\partial I_{mi}^{(l)}(k)} = \sum_{p=1}^{N_{l+1}} \frac{\partial x_{m1}^{(l)}(k)}{\partial I_{mp}^{(l+1)}(k)} \frac{\partial I_{mp}^{(l+1)}(k)}{\partial I_{mi}^{(l)}(k)} \\ \frac{\partial I_{mp}^{(l+1)}(k)}{\partial I_{mi}^{(l)}(k)} &= \frac{\partial I_{mp}^{(l+1)}(k)}{\partial x_{mi}^{(l)}(k)} \frac{\partial x_{mi}^{(l)}(k)}{\partial I_{mi}^{(l)}(k)} = w_{pi}^{(l+1)}(k) x_{mi}^{(l)}(k) (1 - x_{mi}^{(l)}(k)) \\ \frac{\partial x_{m1}^{(l)}(k)}{\partial I_{mp}^{(l+1)}(k)} &= \delta_{mp}^{(l+1)}(k) \end{aligned}$$

$$\delta_{mi}^{(l)}(k) = \sum_{p=1}^{N_{l+1}} \delta_{mp}^{(l+1)}(k) w_{pi}^{(l+1)}(k) x_{mi}^{(l)}(k) (1 - x_{mi}^{(l)}(k)), i = 1 \sim N_l \quad (2-40)$$

这样由  $\delta_{m1}^{(L)}(k)$  可求得  $\delta_{mi}^{(L-1)}(k), i = 1 \sim N_{L-1}$ , 再由  $\delta_{mi}^{(L-1)}(k)$  求  $\delta_{mi}^{(L-2)}(k)$  等等直至  $\delta_{mi}^{(1)}(k), i = 1 \sim N_1$ 。

这样,  $w_{ij}^{(l)}(k)$  的计算已解决。由于  $\delta_{mi}^{(l)}(k)$  是由输出层 ( $l = L$ ) 反推至第 1 隐层 ( $l = 1$ ) 来计算的, 所以此学习算法称为 BP 算法。由于学习前已采集到全部训练数据, 学习过程可脱离现场且每一个权调整递推计算节拍需要调用训练集的全部数据, 故称为离线批处理 BP 算法。

### 2.4.3 离线随机梯度 BP 算法

与前述批处理算法的区别是每个权调整节拍  $k$  只从训练集中取出一组训练数据 (依次取或随机取) 记为  $\mathbf{X}(k), \hat{y}(k)$ 。相应的权调整量计算公式是

$$\Delta w_{ij}^{(l)}(k) = 2\alpha(k) [\hat{y}(k) - x_1^{(L)}(k)] \left. \frac{\partial x_1^{(L)}}{\partial w_{ij}^{(l)}} \right|_{\xi=\xi(k)} \quad (2-41)$$

其中  $x_1^{(L)}(k)$  可由前向计算得到, 式中的偏微分可用式 (2-38) ~ 式 (2-40) 计算。 $\alpha(k)$  可按式 (2-23) 规定的条件选择, 此时随着  $k$  的增大  $\xi(k)$  将趋向于  $R_{\text{emp}}(\xi)$  的一个局部极小点。

另一种方法是每个节拍  $k$  从训练集中取出若干组数据 (例如 10 组) 对网络进行训练。

### 2.4.4 在线 BP 算法

与离线方法的区别是实时实地采集训练数据并对网络进行训练, 从而使网络特性自适应于变化的环境。设按离散时间  $n$  (实时) 采集到的训练数据是  $(\mathbf{X}(n), \hat{y}(n))$  那么实时经验风险可定义如下:

$$R_{\text{emp}}(\xi(n)) = \frac{1}{N} \sum_{\nu=n-N}^{n-1} (\hat{y}(\nu) - x_1^{(L)}(\nu))^2 \quad (2-42)$$

其中  $\xi(n)$  为随  $n$  而变化的权和阈值向量。  $x_1^{(L)}(\nu)$  是相对于  $\mathbf{X}(\nu)$  的网络输出。 $N$  是取平均

间隔 每个  $n$  对权作一次调整 (调整方法与 2.4.2 节或 2.4.3 节相同);  $N$  越大则取平均的范围越大; 反之, 取平均范围缩小时网络的自适应性加强。

## 2.5 BP 算法的主要问题及其改进

### 2.5.1 BP 算法的主要问题

在实施 BP 算法时, 常遇到下列两种困难。

#### 1. 多个局部极小点和平台问题

$R_{\text{emp}}(\xi)$  作为  $\xi$  的函数一般有多个局部极小点; 有时还会出现平台。图 2-10(a) 和 (b) 给出了  $\xi$  为一维的情况下出现多个局部极小点和平台的示例。按照最陡下降算法, 从任择的一个初值权向量  $\xi(0)$  出发, 经过迭代计算  $\xi$  将收敛到一个局部极小点或者滞留在一个平台上。当输入向量  $x$  的维数  $N$  较大且网络的规模很大时,  $\xi$  的维数很高,  $R_{\text{emp}}(\xi)$  可能存在很多局部极小点和平台, 其中很多不能满足要求 (即其值较全局最小点  $\xi_a$  的值  $R_{\text{emp}}(\xi_a)$  大得多)。为此可以采用下列的解决方案。

(1) 设置多个初值  $\xi(0)$ , 从多个迭代计算结果中选择一个最好的。其缺点是当初值设置过多时计算开销太大。

(2) 采用随机优化算法。

(3) 采用模拟退火算法或遗传算法。

(4) 采用单纯形法。

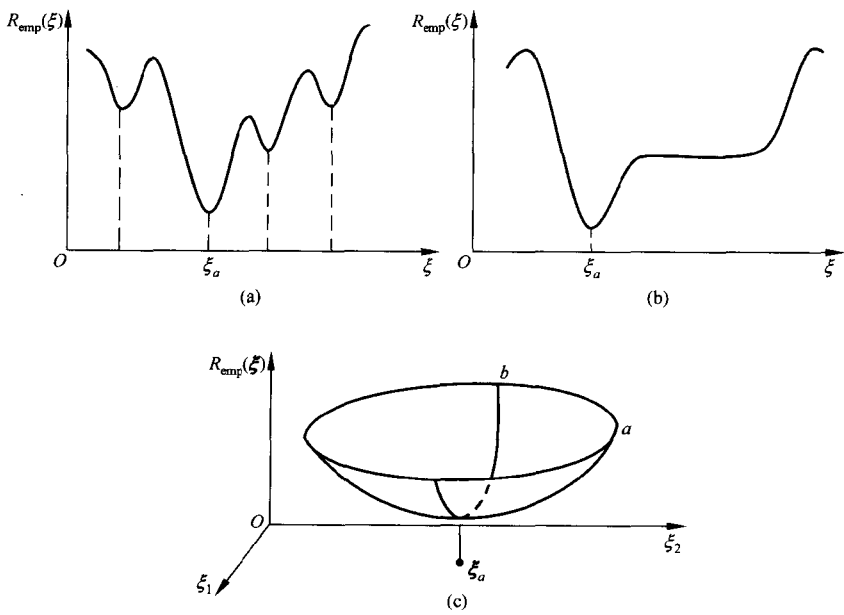


图 2-10  $\xi$  为一维和二维时  $R_{\text{emp}}(\xi)$  的几个示例  
(a) 多个最小点示例; (b) 平台示例; (c) 沟谷的示例

本章将只介绍随机优化算法和单纯形法。

## 2. 步幅 $\alpha$ 的选择问题

若  $\alpha$  选择得太小, 迭代计算收敛速度太慢; 若  $\alpha$  太大, 则在收敛点附近将产生振荡。当在  $\xi$  的不同维上  $R_{\text{emp}}(\xi)$  的下降速度不同时情况更加复杂。当一个极小点附近  $R_{\text{emp}}(\xi)$  的不同维下降速度差异很大时, 即形成所谓沟谷形极小点, 图 2-10(c) 所示是二维空间中的一个例子。由于不同维的下降速度差异很大, 从不同的初值点出发 (例如示例中的  $a$  和  $b$  点) 应采取差异甚大的步幅值。当  $\xi$  的维数甚高时, 矛盾更突出。

为了解决以上两方面困难, 可以采取的途径包括: 改进 BP 算法、非 BP 算法以及二者混合的算法。下面介绍其中的一部分。

### 2.5.2 在 BP 算法中增加惯性

#### 1. 基本惯性算法

式(2-29)中的偏微分用  $S_{ij}^{(l)}(k)$  表示, 即

$$S_{ij}^{(l)}(k) = \left. \frac{\partial R_{\text{emp}}(\xi)}{\partial w_{ij}^{(l)}} \right|_{\xi=\xi(k)}$$

那么惯性 BP 算法的权调整量用下式计算:

$$w_{ij}^{(l)}(k) = -\alpha S_{ij}^{(l)}(k) + \eta \Delta w_{ij}^{(l)}(k-1), \quad k = 0, 1, \dots \text{ 且 } w_{ij}^{(l)}(-1) = 0 \quad (2-43)$$

其中  $\alpha > 0$ , 仍称为步幅;  $1 > \eta > 0$ , 称为惯性系数。如果对上式左右侧施以  $z$  变换, 设  $S_{ij}^{(l)}(k)$  的  $z$  变换  $S_{ij}^{(l)}(z)$  为输入,  $\Delta w_{ij}^{(l)}(k)$  的  $z$  变换  $W_{ij}^{(l)}(z)$  为输出, 则二者关系是

$$\frac{\Delta W_{ij}^{(l)}(z)}{S_{ij}^{(l)}(z)} = \frac{-\alpha}{1 - \eta z^{-1}} \quad (2-44)$$

这是一个 IIR 数字低通滤波器, 它能有效地提高输入信号中的低频分量及降低高频分量, 从而改善迭代计算收敛速度的同时, 有效地抑制收敛点附近的振荡。 $\eta$  值越大, 此效果越强。但是  $\eta$  值不能过大。首先若  $\eta > 1$ , 则此滤波器的极点进入单位圆外, 系统将失去稳定。因此必须保证  $\eta < 1$ 。其次,  $\eta$  越接近于 1, 系统的惯性越大, 其后果是过调越严重。为说明这一点, 将式(2-44)右侧展开为一个无穷级数:  $-\alpha \left\{ \sum_{l=0}^{\infty} \eta^l z^{-l} \right\}$  由此可得

$$\Delta w_{ij}^{(l)}(k) = -\alpha \left\{ \sum_{l=0}^{\infty} \eta^l S_{ij}^{(l)}(k-l) \right\}$$

$\eta$  接近于 1 时, 此级数涉及的取和项很多。当某一节拍  $k$  的  $S_{ij}^{(l)}(k) = 0$  权  $w_{ij}^{(l)}$  应停止调节, 而由于惯性,  $w_{ij}^{(l)}$  不可能立即等于 0, 而是沿原方向调若干步后才止住, 从而造成过调。为避免过调, 可以采取下列的一些方法。

(1) 限制  $\eta$  的最大取值, 例如将其值限制在 0.5 ~ 0.8 之间。当然, 值太小时加惯性项的作用就不大了。

(2) 令  $\alpha$  和  $\eta$  随  $k$  而改变。当  $k$  接近于 0 时  $\alpha(k)$  和  $\eta(k)$  取较大值, 而  $k$  接近于迭代计算终点  $K$  时, 二者取较小值。

(3) 采用 FIR 滤波器或 FIR-IIR 滤波器。

当采用 FIR 滤波器时, 可用下式由  $S_{ij}^{(l)}(k)$  求得  $w_{ij}^{(l)}(k)$



$$\Delta w_{ij}^{(l)}(k) = -\alpha \left\{ \sum_{l=1}^{\mathcal{L}} b_l S_{ij}^{(l)}(k-l) \right\}, \quad k = 0, 1, \dots$$

且

$$S_{ij}^{(l)}(k) = 0, \quad k < 0$$

相应的  $z$  域传输函数是  $\sum_{l=0}^{\mathcal{L}} b_l z^{-l}$ 。这时可以用设计 FIR 低通滤波器的方法求系数  $b_l$ 。

当采用 FIR-IIR 滤波器时，可用下式计算求  $w_{ij}^{(l)}(k)$ ：

$$\Delta w_{ij}^{(l)}(k) = -\alpha \left\{ \sum_{l=1}^{\mathcal{L}} b_l S_{ij}^{(l)}(k-l) \right\} + \eta \left\{ \sum_{l=1}^{\mathcal{F}} a_l \Delta w_{ij}^{(l)}(k-l) \right\}, \quad k = 0, 1, \dots$$

$$S_{ij}^{(l)}(k) = w_{ij}^{(l)}(k) = 0, \quad \text{当 } k < 0$$

其中  $\mathcal{L}$  和  $\mathcal{F}$  是两个正整数。

(2)、(3) 两个方法还可以结合起来使用，即上式中的  $\alpha$  和  $\eta$  成为随  $k$  而变化的函数  $\alpha(k)$  和  $\eta(k)$ 。

## 2. 各个权的步幅和惯性系数独立调节

令各个权具有自身的步幅  $\alpha_{ij}^{(l)}(k)$  和惯性系数  $\eta_{ij}^{(l)}(k)$ ，权调整量按下式计算：

$$\Delta w_{ij}^{(l)}(k) = -\alpha_{ij}^{(l)}(k) S_{ij}^{(l)}(k) + \eta_{ij}^{(l)}(k) \Delta w_{ij}^{(l)}(k-1) \quad k = 0, 1, \dots \quad (2-45)$$

$\alpha_{ij}^{(l)}(k)$  和  $\eta_{ij}^{(l)}(k)$  随迭代计算每一步  $k$  的情况而变化。下面列出的是其中的一种方案所采取的几条规则。

(1) 对于  $k = 0, 1$  设定初值：

$$\alpha_{ij}^{(l)}(0) = \alpha_{ij}^{(l)}(1) = \alpha_0 \quad (\text{例如 } \alpha_0 = 0.2), \quad \forall l, i, j$$

$$\eta_{ij}^{(l)}(0) = \eta_{ij}^{(l)}(1) = \eta_0 \quad (\text{例如 } \eta_0 = 0.7), \quad \forall l, i, j$$

(2) 对于  $k \geq 2$  按下列规则改变  $\alpha_{ij}^{(l)}(k)$  和  $\eta_{ij}^{(l)}(k)$ 。

若  $S_{ij}^{(l)}(k), S_{ij}^{(l)}(k-1), S_{ij}^{(l)}(k-2)$  具有相同符号 则  $\alpha_{ij}^{(l)}(k) = \mu \alpha_{ij}^{(l)}(k-1), \eta_{ij}^{(l)}(k) = \mu \eta_{ij}^{(l)}(k-1)$  (例如  $\mu = 1.1$ )

若  $S_{ij}^{(l)}(k), S_{ij}^{(l)}(k-1), S_{ij}^{(l)}(k-2)$  具有不同符号 则  $\alpha_{ij}^{(l)}(k) = \lambda \alpha_{ij}^{(l)}(k-1), \eta_{ij}^{(l)}(k) = \lambda \eta_{ij}^{(l)}(k-1)$  (例如  $\lambda = 0.5$ )

(3) 设定  $\alpha_{ij}^{(l)}(k)$  和  $\eta_{ij}^{(l)}(k)$  的最大及最小值 例如  $\alpha_{\max} = 0.5, \alpha_{\min} = 0.01; \eta_{\max} = 0.8, \eta_{\min} = 0.1$ 。当超出此范围时即停止增减。

### 2.5.3 修正 BP 算法 (MBP 算法)

用  $S(k)$  表示经验风险函数的梯度，即

$$S(k) = \nabla_{\xi} R_{\text{emp}}(\xi) \big|_{\xi = \xi(k)} \quad (2-46)$$

再定义  $d(k)$  如下：

$$d(k) = \begin{cases} -S(k), & k = 0 \\ -S(k) + \beta_{k-1} d(k-1), & k \geq 1 \end{cases} \quad (2-47)$$

其中

$$\beta_{k-1} = \frac{\|S(k)\|^2}{\|S(k-1)\|^2}$$

参数向量调整按下列迭代公式计算：

$$\begin{aligned}\xi(k+1) &= \xi(k) + \Delta\xi(k) \\ \Delta\xi(k) &= \alpha^* d(k)\end{aligned}\quad (2-48)$$

其中  $\alpha^*$  是一个待定的最佳步幅值, 可以通过下述的线搜索程序将其求得。首先, 将  $(k+1)$  节拍的经验风险表示为  $\alpha$  的函数如下:

$$R_{\text{emp}}(\xi(k+1)) = R_{\text{emp}}(\xi(k) + \alpha d(k))$$

当  $\alpha = \alpha^*$  时, 上式达到最小值。在  $\alpha^*$  附近, 上式可以用一多项式  $(b_0 + b_1\alpha + b_2\alpha^2)$  来近似表示。这样可按以下步骤求  $\alpha^*$ 。第一 求  $\alpha^*$  所处区间。第二 求系数  $b_0, b_1, b_2$ 。第三 通过求近似多项式之最小值求  $\alpha^*$ 。为此, 令  $\alpha$  取一系列值  $\alpha_1, \alpha_2, \alpha_3, \dots$ , 相应的  $\xi(k+1)$  用  $\xi_1(k+1), \xi_2(k+1), \dots$  表示, 则上述一、二两项可按下列程序完成。

(1) 令  $\alpha_1 = 0, \alpha_2 = 1$  若  $R_{\text{emp}}(\xi_2(k+1)) < R_{\text{emp}}(\xi_1(k+1))$  则令  $\alpha_3 = \zeta_1 > 1$  例如  $\zeta_1 = 1.5$  再分两种情况计算:

若  $R_{\text{emp}}(\xi_3(k+1)) > R_{\text{emp}}(\xi_2(k+1))$  则  $\alpha^*$  必然在  $(0, \zeta_1)$  的区间内。通过下列联立方程, 即可由已知的  $\alpha_i$  和  $R_{\text{emp}}(\xi_i(k+1)), i = 1 \sim 3$  求得未知的  $b_0, b_1, b_2$ 。

$$R_{\text{emp}}(\xi_i(k+1)) = b_0 + b_1\alpha_i + b_2\alpha_i^2 \quad i = 1 \sim 3$$

通过求上式对  $\alpha$  的导数为 0 的  $\alpha$  值 即可求得

$$\alpha^* = -b_1/2b_2$$

若  $R_{\text{emp}}(\xi_3(k+1)) < R_{\text{emp}}(\xi_2(k+1))$  则  $\alpha^*$  必然在  $(0, \zeta_1)$  的区间外。这时设  $\alpha_4 = \zeta_2 > \zeta_1$  例如  $\zeta_2 = 2$  若  $R_{\text{emp}}(\xi_4(k+1)) > R_{\text{emp}}(\xi_3(k+1))$  则  $\alpha^*$  必然在  $(1, \zeta_2)$  内; 否则 继续设  $\alpha_5 = \zeta_3 > \zeta_2$  (例如  $\zeta_3 = 2.5$ )。这一过程可继续下去 直至确定  $\alpha^*$  的区间并将其求出。

(2) 令  $\alpha_1 = 0, \alpha_2 = 1$  若  $R_{\text{emp}}(\xi_2(k+1)) > R_{\text{emp}}(\xi_1(k+1))$  则令  $\alpha_3 = \mu_1 < 1$  例如  $\mu_1 = 0.5$ , 再分两种情况:

若  $R_{\text{emp}}(\xi_3(k+1)) < R_{\text{emp}}(\xi_1(k+1))$  则  $\alpha^*$  必然在  $(0, 1)$  区间内 这时仍能由上述的联立方程求得  $b_0, b_1, b_2$  并求得  $\alpha^*$ 。

② 若  $R_{\text{emp}}(\xi_3(k+1)) > R_{\text{emp}}(\xi_1(k+1))$  则可继续设  $\alpha_4 = \mu_2 < \mu_1$  (例如  $\mu_2 = 0.25$ )。如果  $R_{\text{emp}}(\xi_4(k+1)) < R_{\text{emp}}(\xi_1(k+1))$  则  $\alpha^*$  必然在  $(0, \mu_1)$  区间内, 这时仍可通过上述操作求得  $\alpha^*$  反之 若此条件不满足 可以进一步设  $\alpha_5 = \mu_3 < \mu_2$ , 这一过程可继续下去, 直至求得  $\alpha^*$ 。

(3) 为检查  $\alpha^*$  是否确实可用, 判定下列不等式是否成立:

$$R_{\text{emp}}(\xi(k) + \alpha^* d(k)) < R_{\text{emp}}(\xi(k))$$

① 若成立 则  $\alpha^*$  可用。

② 若不成立, 则不可用。这时必须在搜索过程中计算过的  $R_{\text{emp}}(\xi_1(k+1)), R_{\text{emp}}(\xi_2(k+1)), R_{\text{emp}}(\xi_3(k+1)), \dots$  之中找到一个最小的, 设其为  $R_{\text{emp}}(\xi_i(k+1))$  则可令  $\xi(k+1) = \xi_i(k+1)$ 。

上述算法常称为具有线性搜索程序的共轭梯度算法。

#### 2.5.4 随机优化 (RO) 算法

BP 算法、惯性 BP 算法、MBP 算法等基于最陡下降的优化算法都存在着收敛到一个

低质量局部极小点或陷于一个平台上的可能性。RO 算法摆脱最陡下降约束，令搜索方向随机变化，从而可能从一个局部极小点或平台中逸出。下面介绍 RO 算法的一种方案。

(1) 设  $\xi$  的定义域为  $\mathcal{D}$ 。任择一初值参数向量  $\xi(0) \in \mathcal{D}$ 。按节拍  $k$  进行迭代计算，令最大迭代计算次数为  $K$ 。置节拍  $k = 0$ 。

(2) 产生一个维数与  $\xi$  相同的高斯随机向量  $\gamma(k)$ ，其均值为  $b(k)$  (当  $k = 0$  时， $b(k) = 0$ )。若  $(\xi(k) + \gamma(k)) \in \mathcal{D}$  转入(3) 否则转入(4)。

(3) 判断  $R_{\text{emp}}(\xi(k) + \gamma(k)) < R_{\text{emp}}(\xi(k))$  成立否。

若成立，令  $\xi(k+1) = \xi(k) + \gamma(k)$  且令  $b(k+1) = 0.4\gamma(k) + 0.2b(k)$ 。转入(4)。

② 若不成立，而  $R_{\text{emp}}(\xi(k) - \gamma(k)) < R_{\text{emp}}(\xi(k))$  成立，则令  $\xi(k+1) = \xi(k) - \gamma(k)$  而且令  $b(k+1) = -0.4\gamma(k) + b(k)$ 。转入(4)。

若 ①、② 都不成立，令  $\xi(k+1) = \xi(k)$ ，令  $b(k+1) = 0.5b(k)$ 。转入(4)。

(4) 若  $k = K$  结束，若  $k < K$ ，令  $k = k+1$  转入(2)。

## 2.5.5 MBP + RO 的混合算法

此算法中 MBP 算法和 RO 算法交替进行。当判断出 MBP 算法已陷入一个低质极小点或平台时，即转向 RO 算法；当判断出 RO 算法已使  $\xi$  移至一个质量更佳的极小点附近时，即转回 MBP 算法。这样，可以充分发挥定向搜索和随机搜索二者的优势。此过程的示意图如图 2-11 所示。下面讨论混合算法的两个关键问题：如何判断 MBP 算法陷入了一个局部极小点或平台以及判断 RO 算法是否已从一个局部极小点或平台转移到了一个更佳的局部最小点附近。

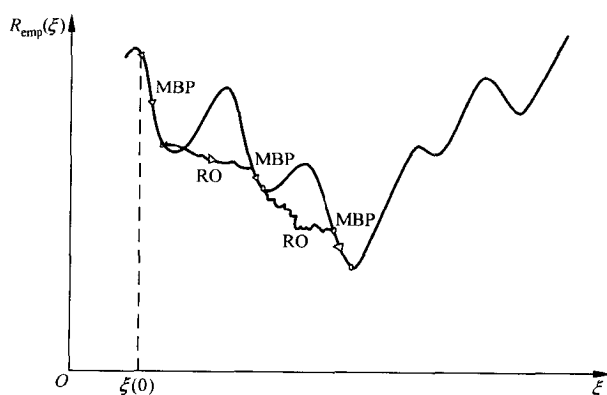


图 2-11 MBP 加 RO 的混合算法

### 1. MBP 算法陷入局部极小或平台的判据

在 MBP 算法中，令  $|\Delta R_{\text{emp}}(\xi(k))| = \delta(k)$ ，它随  $k$  的增加而逐步减小。设  $\epsilon$  是一个小正数（例如  $\epsilon = 0.01$ ），当  $k$  增加到某个值  $k'$  时，如果  $[\delta(k')/R_{\text{emp}}(\xi(k'))] < \epsilon$  且  $R_{\text{emp}}(\xi(k'))$  之值仍较大，即判断已进入一个低质极小点或平台。

### 2. RO 算法从一个局部极小点或平台转移到一个更佳局部最小点附近的判据

RO 算法以  $\xi(k')$  为初值，从  $k = 0$  开始迭代计算。设  $\delta(k)$  是每迭代一步的经验风险下

降量 当  $k$  增加到某个值  $k''$  时, 如果下列不等式成立, 则判断转移成立。

$$\sum_{k=0}^{k''} \delta(k) > \lambda R_{\text{emp}}(\xi(k'))$$

上式左侧是 RC 算法的累计风险下降量 系数  $\lambda$  是一个适当的正参数 (例如  $\lambda = 0.1 \sim 0.2$ )。

混合算法的优点是如果能对  $\xi$  的定义域  $\mathcal{D}$  予以限制 (例如, 每个权的最大绝对值不超过  $10^3$ ) , 则能够收敛到全局最小点。而且收敛速度较快。

## 2.5.6 单纯形法与 BP 算法相结合

单纯形法是一种全局优化算法, 但是当待优化的参数个数较多时其计算效率较低。为此, 可以将其与 BP 算法相结合。如果采用式 (2-46) 的表示式, 那么带惯性项的权向量调整量可以用下式来计算:

$$\Delta \xi(k) = -\alpha^*(k) S(k) + \eta^*(k) \Delta \xi(k-1), \quad k = 0, 1, 2, \dots \quad (2-49)$$

其中  $\xi(-1) = 0$ ;  $\alpha^*(k)$  和  $\eta^*(k)$  是两个待定系数, 它们使  $R_{\text{emp}}(\xi(k+1)) = R_{\text{emp}}(\xi(k) + \Delta \xi(k))$  达到极小值。这样, 对第  $k$  步递推计算而言, 所需解决的是只有两个待优化参数—— $\alpha(k), \eta(k)$ ——的优化问题。为此, 可以将  $R_{\text{emp}}(\xi(k+1))$  表示为  $\alpha(k)$  和  $\beta(k)$  的函数  $J(\alpha(k), \beta(k))$ 。设  $\mathbf{A} = [\alpha(k), \beta(k)]$  则此函数可表示为  $J(\mathbf{A})$ 。这样, 问题成为在  $\mathbf{A}$  的二维空间 (平面) 中找一个点  $\mathbf{A}^* = [\alpha^*(k), \beta^*(k)] = \text{argmin} J(\mathbf{A})$ 。

用单纯形法求  $\mathbf{A}^*$  的程序如下 (参见图 2-12)。首先, 在  $\alpha(k) - \eta(k)$  平面中任择 3 个点  $L, M, H$  作为求  $\mathbf{A}^*$  的计算初值。假设  $J(L) < J(M) < J(H)$ 。按照下式用这 3 个点计算出另外 6 个点:  $F, R, E, S', S'', D$ 。

$$F = \frac{1}{2}(L + M), \quad R = F + (F - H)$$

$$E = R + (F - H), \quad S' = \frac{1}{2}(F + R)$$

$$S'' = \frac{1}{2}(H + F), \quad D = \frac{1}{2}(H + L)$$

然后, 按下列程序求 3 个新点;

(1) 若  $J(E) < J(R) < J(L)$  则以  $E, L, M$  为 3 个新点 反之 转入 (2)。

(2) 若  $J(R) < J(E) < J(L)$  或  $J(L) < J(R) < J(M)$  则以  $R, L, M$  为 3 个新点; 反之 转入 (3)。

(3) 若  $J(M) < J(R) < J(H)$  则以  $S', L, M$  为 3 个新点 反之 转入 (4)。

(4) 若  $J(H) < J(R)$  且  $J(H) > J(S'')$  则以  $S'', L, M$  为 3 个新点 反之 转入 (5)。

(5) 以  $D, F, L$  为 3 个新点。

以上计算递推进行, 直至  $(J(H) - J(L))/J(L)$  小于某个门限值  $\epsilon_j$  为止 ( $\epsilon_j \ll 1$ )。接着, 按照式 (2-49) 做下一步计算。这个方案利用单纯形法在待优化参数个数较少时效率高的优点, 且单纯形计算中无需计算偏导数, 因而计算量较低。它能在很大的范围内搜索,

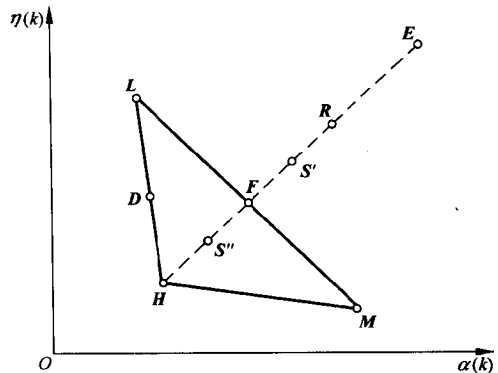


图 2-12 单纯形法求  $\mathbf{A}^*$  的程序

易于脱离低质局部极小点或平台。与 MBP 算法相比，它有 2 个搜索变量而 MBP 只有一个，因此搜索范围和效率更高。若干模拟实验结果证实了以上论断，见文献[6]。

如果将式 (2-49) 右侧第 2 项中的  $\xi(k-1)$  换成  $\gamma(k)$  后者是一个维数与  $\xi(k-1)$  相同的高斯随机向量，那么可以形成定向搜索和随机搜索相结合的单纯形算法。

### 2.5.7 改善学习效果的一些简单策略

#### 1. 输入向量和输出向量规格化

设输入向量  $\mathbf{X}$  的第  $j$  分量  $x_j$  的取值范围是  $(x_{j\min}, x_{j\max})$  当不同分量的取值范围相差甚大时，网络的学习精度和速度都因受其影响而降低。为此可以对每个分量  $x_j$  作线性变换得到  $x'_j$ ， $x'_j = a_j x_j + b_j$  其取值范围是  $(x'_{j\min}, x'_{j\max})$ 。如果  $a_j$  和  $b_j$  按下式计算，

$$a_j = \frac{2}{x_{j\max} - x_{j\min}}, \quad b_j = 1 - a_j x_{j\max} \quad (2-50)$$

那么对于所有分量皆有  $x'_{j\min} = -1, x'_{j\max} = 1$ 。这样，可以用  $\mathbf{X}' = [x'_1, \dots, x'_N]$  取代  $\mathbf{X}$  作为网络的输入向量，它的每一个分量的取值范围都是  $(-1, 1)$ 。对输出向量也可作类似处理。

#### 2. 在批处理算法和随机梯度算法之间进行选择

对于一个具体问题，哪种算法更好一些？这一点难下定论。有些问题的模拟实验结果表明，随机梯度算法的学习效率（收敛到高质局部极小点的速度）比批处理算法高。其中步幅函数  $\alpha(k)$  取阶梯下降形式，如图 2-13 所示，即每作  $L$  步调整后  $\alpha(k)$  才下降一个台阶。图中的虚线方程是  $\{A_0 + (k/L)\}^{-1}$ （例如  $A_0 = 1$ ）。

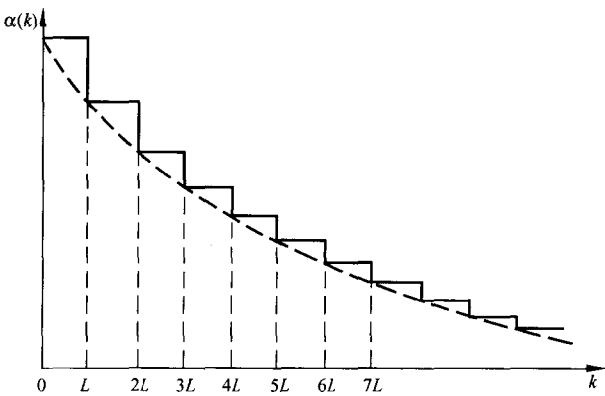


图 2-13 阶梯形步幅函数

## 2.6 RBF 网络

RBF 网络通常取 2 层结构（1 个隐层，1 个输出层）。本节只讨论单输出（MISO）情形，其网络结构与图 2-9 所示结构大致相同（只需去除 2 个为提供阈值参数而设的，输出恒等于 1 的神经元）。如果采用式 (2-7) 和式 (2-8) 的 RBF 形式，输入  $\mathbf{X}$  至输出  $y$  的映射函数是

$$y = \sum_{i=1}^{N_1} w_i^{(2)} (\sqrt{2\pi} \sigma_i^{(1)})^{-1} \exp\left(-\frac{\|\mathbf{X} - \mathbf{W}_i^{(1)}\|^2}{2(\sigma_i^{(1)})^2}\right) \quad (2-51)$$

其中隐层取 RBF，输出层取线性函数； $\mathbf{W}_i^{(1)}$  是输入至隐层第  $i$  神经元的权向量，其维数与  $\mathbf{X}$  相同，皆为  $N$ ；隐层含  $N_1$  个神经元， $w_i^{(2)}$  是其第  $i$  神经元至输出的权值。下面讨论其学习算法以及若干应用特点。

### 2.6.1 RBF 网络的学习算法

设训练集为  $(\mathbf{y}_m, \mathbf{X}_m), m = 1 \sim M$ ，则权参数的学习分下列三个阶段进行。

(1) 隐层各神经元输入权向量  $\mathbf{W}_i^{(1)}$  的学习。这一阶段采用自组织学习算法，将  $\mathbf{X}_m, m = 1 \sim M$  分为  $N_1$  个聚类区，并从中求得  $\mathbf{W}_i^{(1)}$ 。为此可以采用 LBG 算法或 SOM 神经网络学习算法，后者将在第 4 章中作介绍，本节只介绍前者。LBG 算法是用于向量量化 (VQ) 的一种自组织（无监督）聚类学习算法，其目的是求得  $N_1$  个聚类区中心（即  $\mathbf{W}_i^{(1)}, i = 1 \sim N_1$ ），使得平均量化误差为最小。此算法亦常称为修正 Lloyd 算法或 K-平均算法。算法的程序框架如下。

设置最大递推计算步数  $K$  和递推计算中途停止门限  $\delta$ （例如  $K = 10^4, \delta = 0.003$ ）；设置  $N_1$  个权向量初值  $\mathbf{W}_i^{(1)}(0), i = 1 \sim N_1$ （在没有先验知识时可设置其为零均值高斯随机向量。更有效的设置方法见文献 [32]）；设置一个编码误差参数  $D(k)$ ，令  $D(1) = \infty$ ，其中  $k$  为递推计算节拍。令  $k = 0$ 。

以  $\mathbf{W}_i^{(1)}(k), i = 1 \sim N_1$  为中心，将  $\mathbf{X}_1, \dots, \mathbf{X}_M$  划归  $N_1$  个聚类区  $\Omega_i(k), i = 1 \sim N_1$ 。划分准则是：

若

$$\|\mathbf{X}_m - \mathbf{W}_i^{(1)}(k)\| < \|\mathbf{X}_m - \mathbf{W}_j^{(1)}(k)\| \quad \forall j \neq i$$

则

$$\mathbf{X}_m \in \Omega_i(k)$$

这种划分方法称为最近邻划分。

③ 令  $\mathbf{W}_i^{(1)}(k+1)$  等于  $\Omega_i(k)$  内各  $\mathbf{X}_m$  的质心，即

$$\mathbf{W}_i^{(1)}(k+1) = \frac{1}{M_i(k)} \sum_{\mathbf{X}_m \in \Omega_i(k)} \mathbf{X}_m \quad (2-52)$$

其中  $M_i(k)$  为  $\Omega_i(k)$  中包含的  $\mathbf{X}_m$  个数。

④ 计算  $D(k)$  和  $d(k)$ ，计算公式如下：

$$D(k) = \sum_{i=1}^{N_1} \sum_{\mathbf{X}_m \in \Omega_i(k)} \|\mathbf{X}_m - \mathbf{W}_i^{(1)}(k)\|^2$$

$$d(k) = \frac{D(k-1) - D(k)}{D(k-1)}$$

⑤ 判断是否  $d(k) < \delta$ 。若回答为是，转 ⑦；反之，转 ⑥。

⑥ 判断是否  $k < K$ 。若回答为是，令  $k = k+1$ ，转 ⑤；反之，转 ⑦。

⑦ 终止计算。设终止节拍为  $k^*$ ，输出  $\mathbf{W}_i^{(1)}(k^*), i = 1 \sim N_1$ ，作为隐层各神经元的输入权向量，即  $\mathbf{W}_i^{(1)} = \mathbf{W}_i^{(1)}(k^*)$ 。结束。

业已证明，按此程序进行递推计算时  $D(k)$  必然随  $k$  的增加而下降。在终止计算时的编码误差  $D(k^*)$  是否足够小，则取决于初值设置是否恰当。在求得  $\mathbf{W}_i^{(1)}$  后，即可用下式求  $(1)$ ：

$$(\sigma_i^{(1)})^2 = \frac{1}{M_i(k^*)} \sum_{\mathbf{x}_m \in \Omega_i(k^*)} \|\mathbf{x}_m - \mathbf{W}_i^{(1)}\|^2, \quad i = 1 \sim N_1 \quad (2-53)$$

注意，如果采用式 (2-9) 和式 (2-10) 的 RBF 形式由  $\mathbf{X}$  至  $y$  的映射可以表示为

$$y = \sum_{i=1}^{N_1} w_{1i}^{(2)} (2\pi |\boldsymbol{\Sigma}_i^{(1)}|)^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (\mathbf{X} - \mathbf{W}_i^{(1)}) (\boldsymbol{\Sigma}_i^{(1)})^{-1} (\mathbf{X} - \mathbf{W}_i^{(1)})^T \right\} \quad (2-54)$$

其中各  $\mathbf{W}_i^{(1)}$  的求法即可按上述程序进行。各  $\boldsymbol{\Sigma}_i^{(1)}$  则用下列公式计算：

$$\boldsymbol{\Sigma}_i^{(1)} = \frac{1}{M_i(k^*)} \sum_{\mathbf{x}_m \in \Omega_i(k^*)} (\mathbf{x}_m - \mathbf{W}_i^{(1)})^T (\mathbf{x}_m - \mathbf{W}_i^{(1)}), \quad i = 1 \sim N_1 \quad (2-55)$$

当一个聚类区的各维延伸长度相差较大时，后一种表示方法具有更高的准确度。

(2) 在输入至隐层的各  $\mathbf{W}_i^{(1)}$  和  $\sigma_i^{(1)}$  (或  $\boldsymbol{\Sigma}_i^{(1)}$ ) 保持固定的条件下，对于隐层至输出的各个权值  $w_{1i}^{(2)}$  用训练集  $(\hat{y}_m, \mathbf{x}_m), m = 1 \sim M$  按 BP 算法进行有监督的学习。

(3) 对于网络中所有权值，用上述训练集按 BP 算法进行有监督的学习。

上列 (2)、(3) 两项所用的 BP 算法与 2.4 节和 2.5 节所述内容相似，不再赘述。

## 2.6.2 RBF 网络的特点

### 1. RBF 网络和隐层神经元取 Sigmoid 函数的 2 层网络之间的对比

当采用 Sigmoid 函数时，隐层每个神经元对输入向量  $\mathbf{X}$  进行超平面式的划分。当  $\mathbf{X}$  处于由权向量和阈值决定的某个超平面上时，神经元输出等于 0.5；当  $\mathbf{X}$  在超平面一侧距离越远时，神经元输出渐降至 0；在另一侧距离越远时，输出渐增至 1。RBF 网络隐层每个神经元对  $\mathbf{X}$  作超球或超椭球式的划分。对于隐层第  $i$  神经元，超球中心为  $\mathbf{W}_i^{(1)}$ 。当  $\mathbf{X} = \mathbf{W}_i^{(1)}$  时，该神经元输出达最大值；当  $\mathbf{X}$  与  $\mathbf{W}_i^{(1)}$  的欧氏距离增大时，输出趋向于 0 且  $\sigma_i^{(1)}$  值越小时趋 0 速度越快。由于隐层每个神经元只对输入向量空间的某个局部区域中的  $\mathbf{X}$  作出响应，所以常称之为局部接收场网络。

### 2. RBF 网络的学习特点

RBF 网络的一个突出优点是学习速度比采用 Sigmoid 函数的网络快得多。它的第一阶段学习 (LBG 算法或下述 SOFM 算法) 只对隐层神经元进行训练，速度快。第二阶段学习实质上是一个线性函数神经元的 learning 问题，2.2.1 小节已指出，这是一个凸优化问题，即参数空间只有一个最小经验风险函数点，这是一个易于学习的情况。第三阶段学习只是在前两阶段基础上对权值作精细调整而已。

## 2.6.3 RBF 网络的应用特点

RBF 网络学习快速、高效的优点使其在那些需要实现实时自适应的系统（例如通信系统）中应用广泛<sup>[33]</sup>。当代的数字通信系统，特别是数字移动通信系统，对于自适应信道

均衡器提出了很高要求，它用来抑制由于多径效应、传输畸变等原因造成的符号间干扰并且抑制各种外来噪声和干扰。现有的各种信道自适应均衡器实现方案中，有些算法可以在理论上证明是最优的（例如最大似然 Viterbi 算法），但是因计算过于复杂而难以实现；有些算法十分简单（如横向滤波器）但是性能欠佳。用 RBF 神经网络实现信道自适应均衡器时，可以用较简单的算法获得较优的性能，从而成为一种优选方案。图 2-14 显示了一个横向滤波器型信道均衡器和一个 RBF 网络信道均衡器的构成。

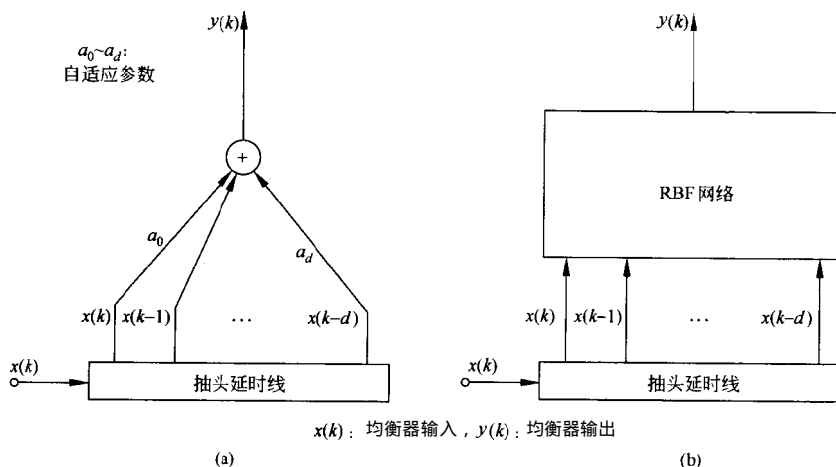


图 2-14 横向滤波器均衡器和 RBF 网络均衡器  
(a) 横向滤波器均衡器；(b) RBF 网络均衡器

RBF 神经网络实现的是非线性映射（横向滤波器只能作线性映射），它集中了聚类、最陡下降等算法，具有鲁棒性（robust）<sup>[34]</sup>。此外，在性能上，它与统计意义上最佳的贝叶斯（Bayes）均衡器十分接近。从计算公式看，二者几乎是相同的（见文献<sup>[33]</sup>）。将 RBF 网络用于判决反馈均衡器（DFE）时，一些模拟实验结果表明，其性能甚至优于采用最大似然 Viterbi 算法的均衡器<sup>[33]</sup>。因此 RBF 网络成了促进通信、神经网络、数字信号处理三者相结合的一个关键点。RBF 还可以用于信号处理的许多领域，如时间序列预测<sup>[35]</sup>、系统建模<sup>[36]</sup>、干扰抑制<sup>[36]</sup>和各种信道均衡问题<sup>[38,42]</sup>。

现在从理论上已证明，只要隐层神经元数足够多，RBF 神经网络可以以任意精度逼近任何单值连续函数。当采用式（2-7）和式（2-8）给出的高斯型函数时，证明方法见文献<sup>[39]</sup>。当采用其他径向基函数时的证明方法见文献<sup>[40]、[41]</sup>。

## 2.7 小波神经网络

小波（wavelet）分析作为一种重要的函数逼近工具应用于人工神经网络，近年来颇受重视<sup>[48~51]</sup>。小波分析的基本概念和算法可以参考文献<sup>[52]、[54]</sup>。如何在人工神经网络中应用小波分析，有多种方案<sup>[48]</sup>。下面只举例说明其中的一种。



### 2.7.1 用小波网络实现映射 $y = f(x)$ 的分析

小波分析的出发点是基本小波尺度函数  $\varphi(x)$ ,  $x$  为实数, 通过下式给出的扩张和移位运算, 可以形成一系列小波  $\varphi_{l,i}(x)$ 。 $\varphi_{l,i}(x)$  的定义为

$$\varphi_{l,i}(x) = 2^{l/2} \varphi(2^l x - i) \quad (2-56)$$

其中  $l, i$  为整数。设实数空间  $\mathbb{R}$  上的所有平方可积函数构成空间  $L^2(\mathbb{R})$  那么只要  $\varphi(x)$  选择恰当, 式 (2-56) 给出的小波系列形成  $L^2(\mathbb{R})$  的正交归一基。它的一项特别重要性质是, 对于任何  $\hat{f}(x) \in L^2(\mathbb{R})$ , 可以用下列函数对其均匀或均方逼近:

$$\hat{f}(x) = \sum \langle \hat{f}, \varphi_{L,i} \rangle \varphi_{L,i}(x) \quad (2-57)$$

其中  $\langle f, \varphi_{L,i} \rangle$  表示  $f(x)$  和  $\varphi_{L,i}(x)$  的内积 取和范围包括  $\langle f, \varphi_{L,i} \rangle$  为非 0 的  $i$  值。以均方逼近为例, 对于任何  $\epsilon > 0$  只要  $L$  足够大, 下式即成立

$$\left\| \hat{f}(x) - \sum \langle \hat{f}, \varphi_{L,i} \rangle \varphi_{L,i}(x) \right\|_{L^2} < \epsilon \quad (2-58)$$

其中  $\| \cdot \|_{L^2}$  表示  $L^2$  模, 见式 (2-14)。

可以采用的小波函数有多种<sup>[55]</sup>。下面给出典型的 Lemarie-Meyer 小波尺度函数作为一个示例。它的时域函数  $\varphi(x)$  和对应的频域函数  $\Phi(\omega)$  (即  $\varphi(x)$  的傅里叶变换) 在下面给出。其图形如图 2-15 所示。

$$\varphi(x) = \frac{2 \sin \frac{2\pi}{3} x}{x} - \frac{\cos \frac{2\pi}{3} \left( x - \frac{3}{4} \right)}{x - \frac{3}{4}} + \frac{\cos \frac{4\pi}{3} \left( x - \frac{3}{4} \right)}{x - \frac{3}{4}} - \frac{\cos \frac{4\pi}{3} \left( x + \frac{3}{4} \right)}{x + \frac{3}{4}} + \frac{\cos \frac{2\pi}{3} \left( x + \frac{3}{4} \right)}{x + \frac{3}{4}} \quad (2-59)$$

$$\Phi(\omega) = \begin{cases} 1, & |\omega| \leq \frac{2\pi}{3} \\ \sin \frac{3\pi}{4}, & \frac{2\pi}{3} \leq |\omega| \leq \frac{4\pi}{3} \\ 0, & |\omega| \geq \frac{4\pi}{3} \end{cases} \quad (2-60)$$

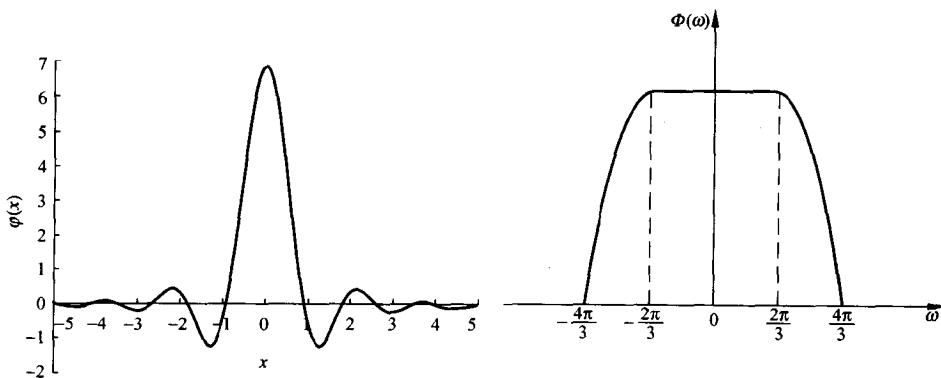


图 2-15 函数  $\varphi(x)$  和  $\Phi(\omega)$

下面首先讨论  $y = f(x)$  的小波神经网络实现，然后讨论  $y = \hat{f}(\mathbf{X})$  的实现  $\mathbf{X} \in \mathbb{R}^N$ ,  $N > 1$ 。

## 2.7.2 $y = f(x)$ 的小波神经网络实现

对于给定的  $L$  值，小波神经网络的映射函数可按照式 (2-57) 表示为

$$\left. \begin{aligned} y = f(x) &= \sum_i w_i \varphi_{L,i}(x) \\ w_i &= \langle \hat{f}, \varphi_{L,i} \rangle \end{aligned} \right\} \quad (2-61)$$

其结构如图 2-16 所示。

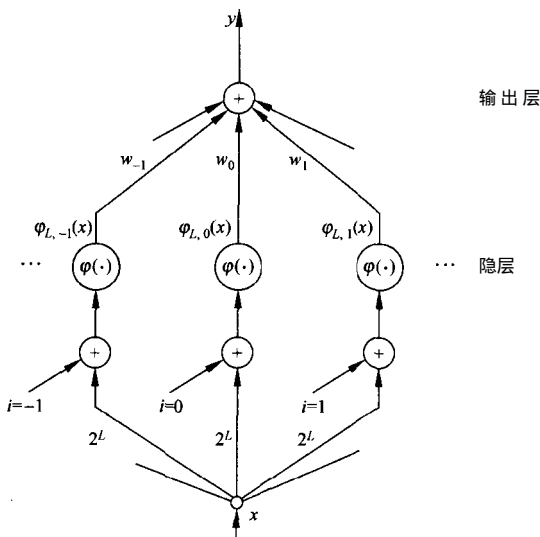


图 2-16  $y = f(x)$  的小波神经网络的实现结构图

当被逼近函数  $f(x)$  未知时，只能由一个训练集  $(\hat{y}_m, x_m), m = 1 \sim M$  求各  $w_i$ 。即求各  $w_i$  使得  $R_{\text{emp}}(\mathbf{W}) = \frac{1}{M} \sum_{m=1}^M (\hat{y}_m - y_m)^2$  达到最小，其中  $y_m = \sum w_i \varphi_{L,i}(x_m)$ ， $\mathbf{W}$  是各  $w_i$  构成的向量。由于  $\varphi_{L,i}(\cdot)$  是一固定函数，所以上述问题归结为求最佳  $\mathbf{W}$ ，使得  $\dots \varphi_{L,-1}(x), \varphi_{L,0}(x), \varphi_{L,1}(x), \dots$  的线性组合求得的  $y$  与理想值  $\hat{y}$  之间的均方误差达到最小。2.2 ~ 2.5 节中已给出了这个问题的完整求解算法。下面讨论用小波神经网络实现一个映射任务时的实施途径。

首先将  $x$  的取值范围通过规格化调整为  $(-1/2, 1/2)$  (见 2.5.7 小节)。对于固定的  $L$  值，可以用下述方法确定隐层的神经元个数；其中每个神经元的输入为  $x$ ，输出为  $\varphi_{L,i}(x)$  见式 (2-61) 和图 2-16。由于每个神经元做的是小波运算，也可称之为小波元。注意相邻两个小波元函数  $\varphi_{L,i-1}(x)$  和  $\varphi_{L,i}(x)$  对  $x$  的操作具有相同的形式，只是间隔开  $\Delta x = 2^{-L}$ ，因此在  $(-1/2, 1/2)$  区间内只需安排  $2^L + 1$  个小波元。考虑到左右边缘的效应，在  $-1/2$  和  $1/2$  之外可以再各安排  $P/2$  个小波元 (例如  $P = 2$  或 4)。这样，隐层共需

$2^L + P + 1$  个神经元 (小波元)。图 2-17 显示了  $L = 2$  及  $P = 2$  时隐层 7 个神经元的函数  $\varphi_{L,i}(x)$  的中心点；相应的映射函数为

$$y = f(x) = \sum_{i=-3}^3 w_i \varphi_{L,i}(x)$$

小波函数中的参数  $L$  可通过尝试来决定。即首先选一个偏小的  $L$  值，计算出相应网络的  $R_{\text{emp}}(\mathbf{W})$ 。如果  $R_{\text{emp}}(\mathbf{W})$  不能满足要求，则令  $L$  逐渐增加，直至得到满足要求的结果为止。

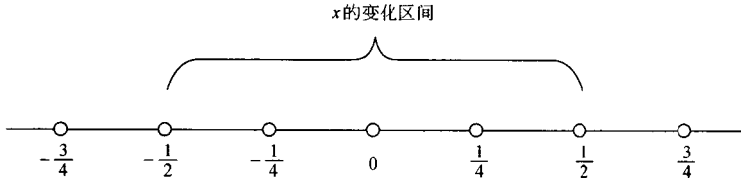


图 2-17  $L = 2, P = 2$  时  $\varphi_{L,i}(x)$  的中心点

### 2.7.3 $y = f(\mathbf{X})$ 的小波神经网络实现

设  $\mathbf{X} = [x_1, \dots, x_N], N > 1$ ，各小波元函数可表示为

$$\varphi_{L,I}(\mathbf{X}) = \prod_{n=1}^N \varphi_{L,i_n}(x_n) \quad (2-62)$$

其中  $I = [i_1, \dots, i_N]$ 。对于给定的  $L$  值，可实现的映射函数是

$$y = f(\mathbf{X}) = \sum_I w_I \varphi_{L,I}(\mathbf{X}) \quad (2-63)$$

实施途径的第一步是将每个分量  $x_n$  的取值范围通过规格化调整为  $(-\frac{1}{2}, \frac{1}{2})$ 。接着，对于固定的  $L$  值，每个  $i_n$  的取值为  $(-\frac{p}{2} - 2^{L-1} \sim 2^{L-1} + \frac{p}{2})$ ，即总共有  $2^L + p + 1$  个取值；相应地， $I$  共有  $(2^L + p + 1)^N$  个取值，这就是网络隐层的神经元 (小波元) 个数。相应地有  $(2^L + p + 1)^N$  个小波函数，其中心构成  $N$  维超立方网格。最后，求各  $w_I$  的方法与上一小节相同。

此方案的问题是当  $N$  较大时，隐层神经元数大得惊人。例如当  $N = 10$ ，取  $L = 3, p = 2$  时，约需  $2.6 \times 10^{10}$  个小波元，而且为了训练这个网络所需的训练样本数更是大得惊人<sup>[48]</sup>。为了解决这个问题，必须通过某种途径将  $\mathbf{X}$  的维数压缩得足够小。下面列举几种方法。

(1) 采用主值分量分析 (principle-component analysis, PCA)。可以通过 PCA 将  $\mathbf{X}$  转换为  $Q$  维向量  $\mathbf{Z} = [z_1, \dots, z_Q], Q < N$ 。转换公式是

$$\mathbf{Z} = \mathbf{X}\Psi \quad (2-64)$$

其中  $\Psi$  是一个  $Q \times N$  维矩阵，它的  $Q$  个列向量等于训练集中  $\mathbf{X}_m, m = 1 \sim M$  的相关阵前  $Q$  个最大特征值相应的特征向量 (PCA 可以用线性 MLFN 来实现，见 2.8 节)。这样，式 (2-63) 可以用下式替代：

$$y = f(\mathbf{X}) = f_1(\mathbf{X}\Psi) = \sum_I w_I \varphi_{L,I}(\mathbf{Z}) \quad (2-65)$$

$$\varphi_{L,I}(\mathbf{Z}) = \prod_{n=1}^Q \varphi_{L,i_n}(z_n)$$

其中  $\mathbf{I} = [i_1, \dots, i_Q]$ 。如果训练集中的各个  $\mathbf{X}_m$  具有近似于高斯型的概率分布函数且只有少数几个特征值较其他特征值大许多时， $Q$  值可以压得很低。

(2) 采用 VQ 方法将  $\mathbf{X}$  的全部定义域分为若干个小聚类区，每个区用一单独的小波神经网络实现输入至输出的映射。由于每个小区中  $\mathbf{X}_m$  的分布较简单，因而通过变换后可将维数  $Q$  压得很低。

(3) 采用 PPR (projection pursuit regression) 可以将一个映射函数  $y = f(\mathbf{X})$  表示成

$$y = f(\mathbf{X}) = \sum_{r=1}^R C_r f_r(\mathbf{X}\Psi_r) \quad (2-66)$$

其中  $\Psi_r$  是一个  $Q_r \times N$  维矩阵且  $Q_r \ll N$  每个  $f_r(\cdot)$  可以用一个小波神经网络来实现。PPR 是一套寻优算法，目的在于求得最佳的  $C_r, \Psi_r, r = 1 \sim R$  使得  $f(\mathbf{X})$  与待实现的映射  $\hat{f}(\mathbf{X})$  之间的均方误差最小<sup>[56,57]</sup>。

## 2.7.4 小波神经网络的特点

(1) 网络隐层的每一个神经元只对输入向量  $\mathbf{x}$  空间的一个局部小区作出响应，这一点与 RBF 网络类似，它们都是局部接收场网络。

(2) 小波神经网络隐层神经元输入权值固定，只有隐层至输出的各个权值由学习来求得。后者是一个线性函数单神经元的学习问题（见 2.2.2 小节），其学习速度非常高。所以小波神经网络的学习效率不仅远高于采用 Sigmoid 函数的 MLFN 而且也高于 RBF 网络。

(3) 当  $\mathbf{x}$  的维数  $N$  较大时，小波神经网络必须采取某种途径进行输入向量维数压缩，这仍是一个在已有基础上有待进一步研究的问题。

## 2.8 MLFN 的前端信号处理

MLFN 的输入  $N$  维向量  $\mathbf{X} = [x_1, \dots, x_N]$  在很多情况下是由一个“时域”函数  $x(t)$  的  $N$  采样值  $x(T), \dots, x(NT)$  构成的，其中  $T$  为采样间隔。这可以表示为  $x_i = x(iT)$ ， $i = 1 \sim N$ 。注意 这里所用的变量  $t$  是一个广义的时间变量，它既可以表示真正的时间，又可以表示任意维欧氏空间中的距离或其他物理量。在用 MLFN 完成函数逼近任务时（例如参数估计、时间序列预测、信道均衡等均属此情况）可以直接以  $\mathbf{X}$  作为网络输入。但是在实现模式识别或分类等任务时，直接以  $\mathbf{X}$  为输入有不利之处，这是因为有关  $\mathbf{X}$  的很多重要信息包含在它的“频域”之中。如将其转换为包含时-频两方面信息的信号后再输入网络，可以提高网络的性能。下面讨论几种实现的方案。

### 2.8.1 短时傅里叶变换 (STFT)

最常用的一种时-频信号表示是一段时域信号  $x(t)$  的 STFT 其计算公式是

$$S_x(f, t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} x(\tau) h(\tau - t) e^{-j2\pi f\tau} d\tau \quad (2-67)$$

其中  $h(t)$  是一时间窗函数，常用的窗函数有汉明（Hamming）窗等，参见文献[58]。在实际使用时，常取  $S_x(f, t)$  的模平方值作为网络输入，并且用  $P_x(f, t)$  表示，即

$$P_x(f, t) = |S_x(f, t)|^2$$

通常称其为“短时谱”。由之可衍生出其他一些时-频信号表示，如短时倒谱、短时听觉加权谱等<sup>[32]</sup>，它们在语音识别等领域中得到广泛应用。实际上，人和动物的听觉及视觉系统前端都作了类似于时-频分析的处理。例如，人的耳蜗管就对输入的声音信号进行了加工，耳蜗的不同部位提取出声音中不同频率分量随时间的变化，再送到高层神经系统作进一步的分析和综合<sup>[32]</sup>

STFT 的时间分辨率  $\Delta t$  正比于窗函数  $h(t)$  主瓣的宽度，而其频率分辨率  $\Delta f$  反比于  $h(t)$  主瓣宽度。因此  $\Delta t \cdot \Delta f$  是一个常数，而且在所有频段上  $\Delta t$  和  $\Delta f$  都取固定值<sup>[54]</sup>。然而对大部分待处理的实际信号而言，不同频段应有不同的时-频分辨率。例如，信号中随时间变化较快部分应具有高时间分辨率（即  $\Delta t$  较小），它所相应的频率分量在高频段，可以具有较低的频率分辨率（即  $\Delta f$  较大）。对于信号中随时间变化较慢部分，情况正好相反。因此，理想的时-频表示方法应该是在保持  $\Delta t \cdot \Delta f$  固定的条件下，令  $\Delta t$  随  $f$  的增加而增加， $\Delta f$  随  $f$  的增加而下降。小波变换是能满足这一要求的一种变换方法。

## 2.8.2 小波变换

对于一个连续时域函数  $x(t)$  其连续小波变换  $CWT_x(\alpha, t)$  定义如下：

$$CWT_x(\alpha, t) = \frac{1}{\sqrt{\alpha}} \int x(\tau) \psi^* \left( \frac{\tau - t}{\alpha} \right) d\tau \quad (2-68)$$

其中  $\psi(t)$  称为基本小波， $*$  表示取共轭。 $\psi(t)$  可以用一个具有低通特性的时域函数  $\varphi(t)$ （例如式(2-59)给出的函数）用频率  $f_0$  调制后得到，

$$\psi(t) = \varphi(t) e^{-j2\pi f_0 t} \quad (2-69)$$

例如， $f_0 = 4/3$ 。 $\alpha$  称为尺度因子，它在频域中的意义恰与频率  $f$  成反比关系。有关小波变换的详细讨论可见参考文献[52]、[54]。

## 2.8.3 WVD 变换

WVD 是 Wigner-Viller distribution 的缩写，WVD 变换给出的时-频表示函数近年颇受研究界重视，见文献[54]，[59]，[60]。一个连续时域函数  $x(t)$  的 WVD 变换记为  $W_x(f, t)$ ，它可以用下式进行计算：

$$W_x(f, t) = \int x\left(t + \frac{\tau}{2}\right) x^*\left(t - \frac{\tau}{2}\right) e^{-j2\pi f\tau} d\tau \quad (2-70)$$

它的时-频分辨率在迄今所知的各种变换中是最高的。可以证明，由 STFT 所得的短时谱可由 WVD 变换通过平滑后获得<sup>[59]</sup> 即有

$$P_x(f, t) = \frac{1}{\pi} \iint W_x(\theta, \tau) W_h(\theta - f, \tau - t) d\theta d\tau \quad (2-71)$$

其中  $W_h(f, t)$  是 STFT 的窗函数  $h(t)$  的 WVD。还可以证明，小波变换可以由 WVD 变换通过平滑计算后获得<sup>[54]</sup> 即有

$$|CWT_x(\alpha, t)|^2 = \frac{1}{2\pi} \iint W_x(\theta, \tau) W_\psi\left(\alpha\theta, \frac{\tau-t}{\alpha}\right) d\theta d\tau \quad (2-72)$$

其中  $W_\psi(f, t)$  是  $\psi(t)$  (见式 2-69)) 的 WVD 变换。可以看到, STFT 小波变换和 WVD 三者密切相关。其中小波变换具有时-频多分辨率的特点, 而且已有一套完善的计算方法, 因此在各种模式识别课题中通过与人工神经网络的结合取得了好结果<sup>[61, 62]</sup>。WVD 变换则具有以下特点。

(1) 对于任何  $x(t)$  即使是复值函数的情况 其 WVD 变换永远是实值函数且可取正、负值。

(2) WVD 变换所造成的时-频平面扩展最小。这就是说它的时间分辨率  $\Delta t$  与频率分辨率  $\Delta f$  之乘积  $\Delta t \Delta f$  在各种变换中是最小的。

(3) 它满足下列两个方程：

$$\int_{-\infty}^{+\infty} W_x(f, t) df = \|x(t)\|^2 \quad (2-73)$$

$$\int_{-\infty}^{+\infty} W_x(f, t) dt = \|X(f)\|^2 \quad (2-74)$$

以上二式中  $\|\cdot\|$  表示取复数的模值。 $X(f)$  是  $x(t)$  的傅里叶变换。

(4) 可以用  $W_x(f, t)$  唯一地恢复原信号  $x(t)$  (最多有一个常数相位因子的差异)。

(5) 它不是一种线性变换。若  $x(t) = x_1(t) + x_2(t)$  则  $x(t)$  的 WVD 变换中除了  $x_1(t)$  和  $x_2(t)$  的 WVD 变换外还包含它们之间的“交叉项”。

WVD 变换与人工神经网络相结合的研究值得重视, WVD 变换在信号检测、模式识别等领域中的应用价值已为若干实例证明<sup>[60]</sup>

## 2.8.4 PCA 变换

压缩一个 MLFN 输入向量  $\mathbf{X}$  的维数 (在保持映射功能的前提下), 既可以改善网络的推广性能 (见 2.11 ~ 2.12 节) 又可以有效地提高其学习效率。特别在用某种时-频变换表示一段输入信号时, 虽然能够更好地给出信号的特征, 但又将信号的维数扩大了很多。例如, 一个时域中的  $N$  维向量  $\mathbf{X} = [x_1, \dots, x_N]$  经过时-频变换后将成为一个  $N \times P$  维时-频矩阵, 它的每一列可以用如下一个列向量来表示:

$$\mathbf{S}(n) = \begin{bmatrix} S(1, n) \\ S(2, n) \\ \vdots \\ S(P, n) \end{bmatrix}, \quad n = 1, \dots, N \quad (2-75)$$

其中  $S(p, n)$  表示某种时-频变换函数,  $p$  是频率维变量,  $n$  是时间维变量。这时输入神经网络的向量维数从  $N$  扩大至  $N \times P$ 。当  $N$  和  $P$  都较大时, 更需要进行有效的压缩。

下面首先介绍用线性 MLFN 实现 PCA 以实现维数压缩的算法, 然后讨论在时-频矩阵压缩中的应用。

### 1. 用线性 MLFN 实现 PCA

设有  $M$  个行向量  $\mathbf{X}_m, m = 1 \sim M$  构成的集合  $\mathbf{X}_m = [x_{m1}, \dots, x_{mN}]$ , 设  $\mathbf{R}$  是一个

$N \times N$  维相关阵，它用下式求得：

$$\mathbf{R} = \sum_{m=1}^M \mathbf{X}_m^T \mathbf{X}_m$$

设  $\mathbf{R}$  的特征值按由大到小的次序排列为  $\lambda_1, \dots, \lambda_N$  设与前  $Q$  个特征值相应的特征向量为  $\mathbf{T}_q, q = 1 \sim Q (Q < N, \mathbf{T}_q$  为列向量)。 $\mathbf{T}_1, \dots, \mathbf{T}_Q$  构成一个  $N \times Q$  维矩阵  $\Psi$  则可由下式

$$\mathbf{Z}_m = \mathbf{X}_m \Psi$$

将  $\mathbf{X}_m$  变换为  $\mathbf{Z}_m = [z_{m1}, \dots, z_{mQ}]$  (参见式 2-64)。只要  $Q$  较  $N$  小很多 通过这一变换即可对  $\mathbf{X}_m$  向量维数实现有效压缩。如果  $\mathbf{X}_1, \dots, \mathbf{X}_M$  只是  $\mathbf{X}$  的全体集合中取出的训练集，那么只有当  $M$  足够大且所取的训练样本有代表性时，这一变换才能对  $\mathbf{X}$  的全体集合有效。这就是 PCA 变换。

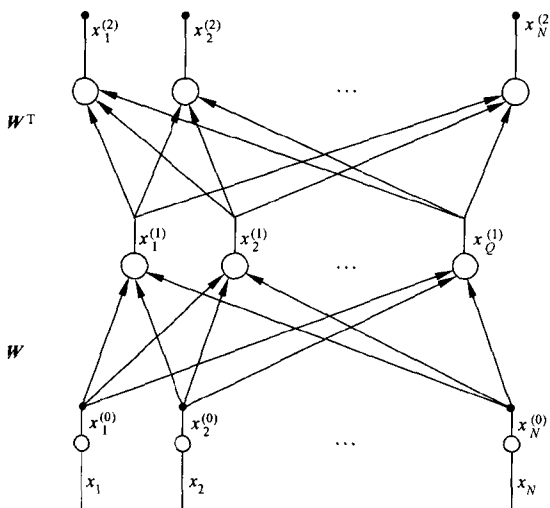


图 2-18 实现 PCA 的 2 层 MLFN 结构图

当  $N$  值较大时，用传统的方法求各  $\lambda_q$  和  $\mathbf{T}_q$  的计算量很大。而采用 2 层线性函数 MLFN 提供了一个求解的有效方法，其结构如图 2-18 所示。网络的输入为  $\mathbf{X} = \mathbf{X}^{(0)} = [x_1^{(0)}, \dots, x_N^{(0)}]$  输出为  $\mathbf{X}^{(2)} = [x_1^{(2)}, \dots, x_N^{(2)}]$  二者维数相同 网络的隐层包含  $Q$  个神经元 其输出可以表示为  $\mathbf{X}^{(1)} = [x_1^{(1)}, \dots, x_Q^{(1)}]$ 。  $Q < N$ 。设  $\mathbf{W}$  是一个  $N \times Q$  维权矩阵，则可通过下列前向计算由  $\mathbf{X}^{(0)}$  求得  $\mathbf{X}^{(2)}$ 。

$$\mathbf{X}^{(1)} = \mathbf{X}^{(0)} \mathbf{W}, \quad \mathbf{X}^{(2)} = \mathbf{X}^{(1)} \mathbf{W}^T \quad (2-76)$$

设  $\mathbf{W}$  的元素是  $w_{ij}, i = 1 \sim N, j = 1 \sim Q$  则上式可以表示为

$$\left. \begin{aligned} x_j^{(1)} &= \sum_{i=1}^N w_{ij} x_i^{(0)}, \quad j = 1 \sim Q \\ x_i^{(2)} &= \sum_{j=1}^Q w_{ij} x_j^{(1)}, \quad i = 1 \sim N \end{aligned} \right\} \quad (2-77)$$

用此 MLFN 求解 PCA 的过程是从训练集  $\mathbf{X}_m, m = 1 \sim M$  中取出向量 当  $\mathbf{X}_m^{(0)} = \mathbf{X}_m$  时，令网络的理想输出为  $\hat{\mathbf{X}}_m^{(2)} = \mathbf{X}_m$  而网络的实际输出为  $\mathbf{X}_m^{(2)} = \mathbf{X}_m^{(0)} \mathbf{W} \mathbf{W}^T$ 。若有一种学习算法能够对  $\mathbf{W}$  进行递推计算，直至收敛到一个最佳  $\mathbf{W}_a$  使得  $\mathbf{X}_m^{(2)}$  与  $\hat{\mathbf{X}}_m^{(2)}$  之间的均方误差达

到最小；同时使  $\mathbf{W}$  的  $Q$  个列向量等于  $\mathbf{R}$  的前  $Q$  个特征值（由大到小排列）相应的特征向量 即使得  $\mathbf{W}_a = \mathbf{\Psi}$ 。下面给出几种学习算法。

(1) 若  $Q = 1$ ，即隐层只含一个神经元，则可以依次或随机地从训练集中取出一个向量 并记为  $\mathbf{X}(k)$ 。令  $\mathbf{X}^{(0)}(k) = \mathbf{X}(k)$  用式 (2-76) 可求得  $\mathbf{X}^{(1)}(k)$  及  $\mathbf{X}^{(2)}(k)$  网络的理想输出为  $\hat{\mathbf{X}}^{(2)}(k) = \mathbf{X}^{(0)}(k)$ 。对网络中各个权  $w_{i1}$  赋予随机初值  $w_{i1}(0)$  后，可按下式进行递推计算 ( $k$  为递推计算节拍,  $k = 0, 1, 2, \dots$ )：

$$w_{i1}(k+1) = w_{i1}(k) + \alpha(k) \{ \hat{x}_i^{(2)}(k) - x_i^{(2)}(k) \} x_1^{(1)}(k) \quad i = 1 \sim N \quad (2-78)$$

注意，式中  $\hat{x}_i^{(2)}(k) = x_i^{(0)}(k) = x_i(k)$ ,  $x_i^{(2)}(k) = w_{i1}(k)x_1^{(1)}(k)$ ,  $x_1^{(1)}(k) =$

$\sum_{i=1}^N w_{i1}(k)x_i^{(0)}(k)$ ;  $\alpha(k)$  是一个步幅函数，它应满足式 (2-23) 的条件。Oja 证明 随着  $k$  趋于无穷大  $[w_{11}, w_{21}, \dots, w_{N1}]^T$  将收敛于  $\mathbf{R}$  最大特征值相应的特征向量<sup>[64]</sup>。式 (2-78) 中 在每一节拍  $k$ ，由隐层惟一神经元至输出层第  $i$  神经元的权调整量  $w_{i1}(k)$  正比于第  $i$  神经元的输出误差与隐层神经元输出值的乘积，这是一种符合 Hebb 学习律的算法。

(2) 若  $1 < Q < N$ ，可按下列公式进行递推计算：

$$w_{ij}(k+1) = w_{ij}(k) + 2\alpha(k) \left\{ \hat{x}_i^{(2)}(k) - \sum_{q=1}^j w_{iq}(k)x_q^{(1)}(k) \right\} x_j^{(1)}(k),$$

$$i = 1 \sim N, \quad j = 1 \sim Q \quad (2-79)$$

式中  $\hat{x}_i^{(2)}(k) = x_i^{(0)}(k)$ ,  $x_j^{(1)}(k) = \sum_{i=1}^N w_{ij}(k)x_i^{(0)}(k)$ ;  $\alpha(k)$  仍是一个满足式 (2-23) 条件的步幅函数。Sanger 证明 当  $k$  趋于无穷大时,  $[w_{1j}, w_{2j}, \dots, w_{Nj}]^T, j = 1 \sim Q$  将趋于  $\mathbf{R}$  的前  $Q$  个最大特征值相应的特征向量<sup>[65]</sup>。式 (2-79) 所采取的算法称为推广 Hebb 学习律。

(3) 若  $1 < Q < N$  文献[63] 给出了下列递推计算公式，它可达到与 (2) 相同的收敛效果：

$$w_{ij}(k+1) = w_{ij}(k) + 2\alpha(k) \{ \hat{x}_i^{(2)}(k) - w_{ij}(k)x_j^{(1)}(k) \} x_j^{(1)}(k) \quad i = 1 \sim N, \quad j = 1 \sim Q \quad (2-80)$$

文献[63] 还讨论了训练集中包含一些非规则样本时，如何采取在目标函数中加非线性修正项来提高其鲁棒性的做法。

(4) 若 1 如果采用最陡下降算法 使节拍  $k$  的下列目标函数  $R_e(\mathbf{W}(k))$  达到最小，

$$R_e(\mathbf{W}(k)) = \sum_{i=1}^N (\hat{x}_i^{(2)}(k) - x_i^{(2)}(k))^2$$

则可以得到下列递推计算公式：

$$w_{ij}(k+1) = w_{ij}(k) + 2\alpha(k) \left[ \{ \hat{x}_i^{(2)}(k) - x_i^{(2)}(k) \} x_j^{(1)}(k) + \sum_{q=1}^N \{ \hat{x}_q^{(2)}(k) - x_q^{(2)}(k) \} w_{qj}(k)x_i^{(0)}(k) \right], \quad i = 1 \sim N, \quad j = 1 \sim Q \quad (2-81)$$

## 2. 用 PCA 实现时-频矩阵的压缩

对于式 (2-75) 的时-频矩阵的  $N$  个列向量  $\mathbf{S}(n), n=1 \sim N$  可求出其  $P \times P$  维相关阵  $\mathbf{R}$  为



$$\mathbf{R} = \sum_{n=1}^N \mathbf{S}(n) \mathbf{S}^T(n)$$

然后用 1 中所述的方法求得  $\mathbf{R}$  的前  $Q$  个特征值相应的特征向量构成的  $P \times Q$  维矩阵  $\Psi$  ( $Q < P$ )。通过下列变换 可将各  $P$  维列向量  $\mathbf{S}(n)$  转换为  $Q$  维列向量  $\zeta(n)$ ：

$$\zeta(n) = \Psi^T \mathbf{S}(n), \quad n = 1 \sim N$$

若  $Q$  较  $P$  小很多, 由  $\zeta(1), \dots, \zeta(N)$  可构成一个比原输入矩阵维数小很多的  $N \times Q$  维时-频矩阵。

## 2.9 MLFN 的函数逼近能力

前文已指出, 对于 2 层单输出 MLFN, 无论隐层取 Sigmoid 函数、小波函数还是 RBF(输出层皆取线性函数), 只要隐层神经元足够多, 就能以任意精度逼近任何有限支单值连续函数。但是, 有一个问题尚待回答: 对于隐层神经元取某一种函数的 MLFN 逼近误差按照什么规律随隐层神经元数的增加而下降? 这与被逼近函数的何种特性有关? 这是一个函数逼近速度的问题, 显然, 对于网络规模的确定是至关重要的。下面针对不同的隐层神经元函数进行讨论。

### 1. 隐层取 Sigmoid 函数时 MLFN 的逼近速度

设被逼近函数为  $y = f(\mathbf{X})$ ,  $\mathbf{X} = [x, \dots, x_N]$ 。2 层单输出且隐层取 Sigmoid 函数、输出层取线性函数的 MLFN 所实现的映射函数可表示为

$$y = f_{N_1}(\mathbf{X}, \xi) = \sum_{i=1}^{N_1} w_{i1}^{(2)} f_s \left[ \sum_{j=1}^N w_{ij}^{(1)} x_j + \theta_i^{(1)} \right] + \theta_1^{(2)} \quad (2-82)$$

其中  $f_s[\cdot]$  表示式 (2-5) 或式 (2-6) 给出的 Sigmoid 函数;  $f_{N_1}(\mathbf{X}, \xi)$  的下标  $N_1$  表示隐层神经元数,  $\xi$  表示所有权和 阈值参数构成的向量。则网络所实现的函数与被逼近函数之间的均方逼近误差为

$$R_{N_1}(\xi) = \int [\hat{f}(\mathbf{X}) - f_{N_1}(\mathbf{X}, \xi)]^2 dP(\mathbf{X}) \quad (2-83)$$

其中  $P(\mathbf{X})$  是  $\mathbf{X}$  的概率分布函数。 $R_{N_1}(\cdot)$  实质上即是式 (2-15) 表示的风险函数 (其中的损失函数  $L(\cdot)$  按照式 (2-12) 来定义), 只是为了明确表示隐层神经元数, 加了下标  $N_1$ 。设  $\xi_0$  使  $R_{N_1}(\xi)$  达到最小值, 即

$$R_{N_1}(\xi_0) = \min R_{N_1}(\xi)$$

$R_{N_1}(\xi_0)$  称为  $N_1$  阶最小均方逼近误差或  $N_1$  阶最小风险, 其值取决于  $N_1$ ,  $P(\mathbf{X})$  和被逼近函数  $\hat{f}(\mathbf{X})$  的性质。下面给出 A. R. Barron 关于此问题的研究所得到的若干结果<sup>[66]</sup>

为了描述被逼近函数  $f(\mathbf{X})$  的特性, 需求得它的傅里叶变换  $F(\omega)$ 。此二者之间的关系由下式描述:

$$\hat{f}(\mathbf{X}) = \int_{\mathbb{R}^N} e^{i\mathbf{X} \cdot \omega} \hat{F}(\omega) d\omega \quad (2-84)$$

其中  $\omega = [\omega_1, \dots, \omega_N]$ ,  $\omega \in \mathbb{R}^N$ 。 $\mathbf{X} \cdot \omega$  表示两个向量的内积 (点积), 为了描述  $\hat{f}(\mathbf{X})$  随  $\mathbf{X}$  而变化的剧烈程度, 可以通过  $\hat{R}(\omega)$  来定义一个参数  $C_{\hat{f}}$  其计算公式如下:

$$C_{\hat{f}} = \int_{R^N} \|\omega\| \|\hat{F}(\omega)\| d\omega \quad (2-85)$$

设  $\hat{f}(\mathbf{X})$  的梯度是  $\nabla_{\mathbf{X}} \hat{f}(\mathbf{X}) = [\partial \hat{f}(\mathbf{X})/\partial x_1, \dots, \partial \hat{f}(\mathbf{X})/\partial x_N]$  易于证明 其傅里叶变换为  $j\omega F(\omega)$ 。可以看到  $C_{\hat{f}}$  是它的模值的积分。 $C_{\hat{f}}$  之值取决于向量  $\mathbf{X}$  和  $\omega$  的维数  $N$  以及  $\hat{f}(\mathbf{X})$  的性质。当  $\hat{f}(\mathbf{X})$  中包含 Kronecker  $\delta$  函数项时,  $\hat{F}(\omega)$  中将包含不随频率  $\omega$  而改变的非 0 常数项 这时  $C_{\hat{f}}$  将趋于无穷大。当  $\hat{f}(\mathbf{X})$  中包含阶跃函数项时,  $\hat{F}(\omega)$  中将包含与  $\|\omega\|^{-1}$  成正比的项, 这时  $C_{\hat{f}}$  也将趋于无穷大。当  $\hat{f}(\mathbf{X})$  为  $S$  阶可导时 ( $S \geq 1$ ),  $\hat{F}(\omega)$  中包含与  $\|\omega\|^{-S-1}$  成正比的项, 这时  $C_{\hat{f}}$  为有限值且  $S$  越大时  $C_{\hat{f}}$  越小;  $S$  越大表明  $\hat{f}(\mathbf{X})$  相对于  $\mathbf{X}$  的变化越平缓。基于以上关于  $C_{\hat{f}}$  的定义, 可有下列定理。

**Barron 定理** [66] 设待逼近的函数  $\hat{f}(\mathbf{X})$  满足下列两点要求: ①  $\mathbf{X}$  的定义域包含原点 即  $\mathbf{X} = \mathbf{0}$  且处于一个半径为  $r$  的超球  $B_r$  之内,  $B_r$  可以表示为  $B_r = \{\mathbf{X}, \|\mathbf{X}\| < r\}$ 。

参数  $C_{\hat{f}}$  为有限值。对于任何  $N_1 \geq 1$ , 总可以在函数集  $f_{N_1}(\mathbf{X}, \xi)$  ( $\xi \in \Delta_{N_1}$  ( $\Delta_{N_1}$  是  $\xi$  的定义域)) 中找一个函数  $f_{N_1}(\mathbf{X}, \xi_0)$  使得下式对任何  $P(\mathbf{X})$  成立:

$$\int_{B_r} [\hat{f}(\mathbf{X}) - f_{N_1}(\mathbf{X}, \xi_0)]^2 dP(\mathbf{X}) \leq \frac{(2r C_{\hat{f}})^2}{N_1} \quad (2-86)$$

此外, 还可以证明下列等式和不等式成立:

$$\theta_1^{(2)} = \hat{f}(\mathbf{0}), \quad \sum_{i=1}^{N_1} |w_{1i}^{(2)}| \leq 2r C_{\hat{f}} \quad (2-87)$$

可以看到, 式 2-86 左侧即是  $R_{N_1}(\xi_0)$ 。

对 Barron 定理的讨论:

(1) 注意, 式 2-86 的成立与  $\mathbf{X}$  取何种概率分布函数  $P(\mathbf{X})$  无关。

(2) 如果通过规格化将  $\mathbf{X}$  的每一维分量的取值范围置为  $(-1, 1)$  则超球  $B_r$  的半径等于  $\sqrt{N}$ 。式 (2-86) 右侧的分子等于  $4NC_{\hat{f}}^2$ 。

(3) 均方逼近误差随着  $N_1$  的增加按  $1/N_1$  的速率下降。这个速率与其他采用固定基函数 (例如多项式函数、样条函数、三角函数等) 的线性组合进行函数逼近时的速率相比要快得多, 后者的下降速率是  $1/N_1^N$  当  $\mathbf{X}$  的维数  $N$  较大时 其降速极慢 这称为“维数的诅咒” (curse of dimensionality)。

## 2. 隐层取 RBF 时 MLFN 的逼近速度

若干研究工作表明 [67, 68] 若  $\hat{f}(\mathbf{X})$  为  $S$  阶可导, 则均方逼近误差随  $N_1$  的增加而下降的速率是  $1/N_1^{\rho}$ ,  $\rho = 1 + N/2S$ 。可以看到 只有  $S \geq N$  时, 均方逼近误差才能以接近于  $1/N_1$  的速率下降。此条件表明 当  $N$  较大时, 只有对于非常“平滑”的函数才能使均方逼近误差下降速率快, 否则难逃维数的诅咒。

## 3. 隐层取小波函数时 MLFN 的逼近速度

若  $\hat{f}(\mathbf{X})$  为  $S$  阶可导 文献 [63] 证明, 小波神经网络的均方逼近误差随  $N_1$  的增加而下降的速度是  $1/N_1^{\frac{1}{\eta}}$ ,  $\eta = N/(S - \frac{1}{2})$ 。只有当  $S$  接近于  $N$  时, 此下降速率才能接近于

$1/N_1$ 。这说明 当输入向量  $\mathbf{X}$  的维数  $N$  较大时，只有对于足够平滑的函数  $\hat{f}(\mathbf{X})$ ，误差下降才足够快。以上讨论表明，对于采用局部接收场的 RBF 或小波网络，为了使逼近误差小于一规定值，当被逼近函数不是非常平滑时，网络规模（即  $N_1$ ）必须随待解决问题的复杂度（即输入向量维数  $N$ ）的增加而指数增加，但是其学习效率很高。而采用全局接收场的 Sigmoid 网络， $N_1$  只需随  $N$  线性增加，它逃脱维数诅咒的代价是学习效率很低。

## 2.10 MLFN 推广能力的统计学习理论

### 2.10.1 学习机与统计学习理论

自然科学研究中最常用的方法之一是归纳推理 (inductive inference)。这就是用有限的因果关系样本（即学习样本或训练样本）得到具有普适性的因果规律。学习机就是通过有限学习样本训练后，给出普适性因果规律的机器。一个通用的学习机模型如图 2-19 所示。它的 3 个模块，G, S, LM 具有下述功能。

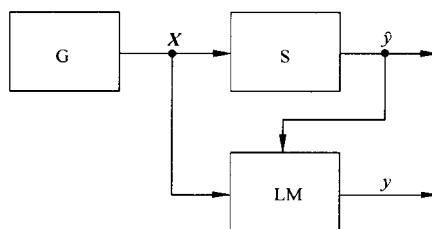


图 2-19 通用学习机模型

G 依次、独立 (或非独立) 地产生随机  $N$  维向量  $\mathbf{X}$  其概率分布函数  $P(\mathbf{X})$  确定但未知。

S: 以  $\mathbf{X}$  作为输入时的理想输出  $\hat{y}$  由 S 提供。S 称为监督器 (教师)。 $\hat{y}$  是一个随机变量，其条件概率分布函数  $P(\hat{y}/\mathbf{X})$  确定但未知。

LM: LM 为学习机，它根据 S 提供的训练样本集  $(\mathbf{X}_m, \hat{y}_m)$ ,  $m = 1 \sim M$  从一组函数  $f(\mathbf{X}, \xi)$ ,  $\xi \in \Lambda$  中选出一个函数  $f(\mathbf{X}, \xi_a)$  使得由式 (2-11) 给出的经验风险  $R_{\text{emp}}(\xi)$  达到最小。这称为经验风险最小归纳原理，记为 ERM(empirical risk minimization)。

基于 ERM 的学习机涉及的诸多问题中，以下两方面对于人工神经网络至关重要。第一，已知对全集合平均的损失是风险函数  $R(\xi)$  (式 (2-15))， $R(\xi_0)$  是其最小值。 $R_{\text{emp}}(\xi)$  是训练集上的平均损失 (式 (2-11))， $R_{\text{emp}}(\xi_a)$  是其最小值。 $\xi_a$  可求而  $\xi_0$  不可求。问题在于：采用  $\xi_a$  时  $R(\xi_a)$  是否足够小？ $R(\xi_a)$  与  $R(\xi_0)$  的差距有多大？第二，当训练集规模  $M$  一定时，应如何选择神经网络的规模以使得  $R(\xi_a)$  达到最小值？第一个问题就是推广问题（用训练集内的有限数量样本求得的  $f(\mathbf{X}, \xi_a)$  是否适用于全体集合，这就是上面的两项提问）。第二个问题称为结构风险最小问题。下面分别对其进行讨论。

### 2.10.2 学习机推广能力的界

为讨论方便 设  $\mathbf{Z} = (\hat{y}, \mathbf{X})$  损失函数  $L(\hat{y}, f(\mathbf{X}, \xi))$  表示为  $Q(\mathbf{Z}, \xi)$  经验风险函数和

风险函数表示如下：

$$R_{\text{emp}}(\xi) = \frac{1}{M} \sum_{m=1}^M Q(Z_m, \xi), \quad \xi \in \Lambda, \quad Z_m = (\hat{y}_m, X_m)$$

$$R(\xi) = \int Q(Z, \xi) dP(Z), \quad \xi \in \Lambda$$

已知  $R_{\text{emp}}(\xi_a) = \min_{\xi} R_{\text{emp}}(\xi)$ ,  $R(\xi_0) = \min_{\xi} R(\xi)$ 。学习机推广能力界的研究在于界定：

(1)  $R(\xi_a)$  与  $R_{\text{emp}}(\xi_a)$  之间的差异。如果二者的差异足够小，则一个学习机通过有限训练样本求得的  $\xi_a$  如能使经验风险足够小，也就能使全集合的风险足够小。

(2)  $R(\xi_a)$  与  $R(\xi_0)$  之间的差异。如果二者足够接近，表明  $\xi_a$  与全集合最优参数  $\xi_0$  足够接近。

Vapnik 和 Chervonenkis 关于统计学习理论的系统研究<sup>[69]</sup> 对于上列两个问题给出了明确的回答。下列定理给出这两个差异的最大界限，即推广能力界。

**Vapnik 和 Chervonenkis 定理** 设  $Q(Z, \xi)$  分成下列两种情况：全有界函数集， $-\infty < A \leq Q(Z, \xi) \leq B < \infty$  全有界非负函数集， $0 \leq Q(Z, \xi) \leq B < \infty$ 。针对这两种情况的推广能力界如下。

(1) 全有界函数集

下列两个不等式成立的概率不小于  $(1 - \eta)$  和  $(1 - 2\eta)$ ,  $0 < \eta \ll 1$ 。

$$R_{\text{emp}}(\xi_a) - \frac{(B - A)}{2} \sqrt{\epsilon} \leq R(\xi_a) \leq R_{\text{emp}}(\xi_a) + \frac{(B - A)}{2} \sqrt{\epsilon} \quad (2-88)$$

$$R(\xi_a) \leq R(\xi_0) + \frac{(B - A)}{2} \sqrt{\epsilon} + (B - A) \sqrt{\frac{-\ln \eta}{2M}} \quad (2-89)$$

二式中的  $\epsilon$  由下式计算， $M$  为训练集规模，其中

$$\epsilon = \frac{4 \left[ h \ln \left( \frac{2M}{h} + 1 \right) - \ln \frac{\eta}{4} \right]}{M}, \quad M > h \quad (2-90)$$

$h$  称为一个学习机的 VCdim (dim 是 dimension 的缩写) 其计算方法见 2.10.3 小节。

(2) 全有界非负函数集

下列两个不等式成立的概率不小于  $(1 - \eta)$  和  $(1 - 2\eta)$ ,  $0 < \eta \ll 1$ 。

$$R(\xi_a) \leq R_{\text{emp}}(\xi_a) + \frac{B\epsilon}{2} \left( 1 + \sqrt{1 + \frac{4R_{\text{emp}}(\xi_a)}{B\epsilon}} \right) \quad (2-91)$$

$$R(\xi_a) \leq R(\xi_0) + \frac{B\epsilon}{2} \left( 1 + \sqrt{1 + \frac{4}{\epsilon}} \right) + B \sqrt{\frac{-\ln \eta}{2M}} \quad (2-92)$$

其中  $\epsilon$  仍用式 (2-90) 计算。

应注意，此定理给出的推广界与概率分布  $P(Z)$  无关。可以看到，当  $h$  一定而训练集的规模不受限制时，此定理表明，当  $M$  趋向无穷大时， $R(\xi_a)$  既趋近于  $R_{\text{emp}}(\xi_a)$  又趋近于  $R(\xi_0)$ 。

## 2.10.3 VCdim 的计算

### 1. 指示函数集情况

对于一个二分类模式识别问题,对于任一输入  $X$  理想输出  $\hat{y}$  只能等于 1 或 0 分别表示一种类别;相应地,  $f(X, \xi)$  也只能等于 1 或 0。当  $\hat{y}$  与  $f(X, \xi)$  一致时,分类正确,定义其损失函数为  $Q(Z, \xi) = L(\hat{y}, f(X, \xi)) = 0$  反之,当二者不一致,即分类错误时,定义其损失函数  $Q(Z, \xi) = 1$ 。这种损失函数集称为指示函数集。一个指示函数集的 VCdim 可用下述方法计算<sup>[69]</sup>。

设  $Z_1, \dots, Z_r$  是  $Z$  的任意  $r$  个取值。对于任何  $\xi \in \Delta$  设  $Q(Z_1, \xi) = \delta_1, \dots, Q(Z_r, \xi) = \delta_r$ 。  $[\delta_1, \dots, \delta_r]$  构成一个  $r$  维二进制向量,它表示取参数  $\xi$  时,  $Q(Z, \xi)$  对  $Z_1, \dots, Z_r$  所作的一种划分(即有的分类正确、有的分类错误)。可能出现的不同类型划分总共有  $2^r$  种。先设  $r = 1$  如果对于任何可能出现的  $Z_1, \dots, Z_r$  通过在  $\Delta$  内改变  $\xi$  即可实现所有  $2^r$  种划分,那么令  $r$  增加 1。接着,重复上述检验,如得以通过则令  $r$  再增加 1。这一过程继续下去,直到  $r$  增至某值为止,当超过此值时,无论  $\xi$  如何改变也不可能实现全部  $2^r$  种划分。该值即定为此函数集的 VCdim。

### 2. 实函数集情况

设  $Q(Z, \xi), \xi \in \Delta$  是一个实函数集。首先,将其转换成一个指示函数集  $I(Z, \xi, \beta)$  如下所示:

$$I(Z, \xi, \beta) = \text{sgn}[Q(Z, \xi) - \beta], \quad \xi \in \Delta, \quad \beta \in R$$

其中  $\text{sgn}[\cdot]$  即是式 (2-4) 的硬限幅函数,  $R$  为实数域。 $Q(Z, \xi)$  的 VCdim 即等于  $I(Z, \xi, \beta)$  的 VCdim (后者的参数变化域包括  $\Delta$  和  $\beta$ )

### 3. VCdim 计算实例之一 —— 线性指示函数集

此函数集可以表示如下:

$$Q(Z, \xi) = \text{sgn}\left[\theta + \sum_{i=1}^N w_i z_i\right] \quad (2-93)$$

其中  $Z = [z_1, \dots, z_N]$ ,  $\xi = [\theta, w_1, \dots, w_N]$ 。可以证明,该函数集的 VCdim 计算公式如下:

$$h = N + 1 \quad (2-94)$$

这里只证明  $N = 2$  的情况;对于  $N > 2$  的情况,可用类似方法予以证明。当  $N = 2$  时,  $Z = [z_1, z_2]$ 。现在,略过  $r = 1$  和  $r = 2$ , 直接证明当  $r = 3$  时,对于  $R^2$  (平面)上的任意 3 个  $Z$  的取值  $Z_1, Z_2, Z_3$  用式 (2-93) 的函数集可以实现  $2^r = 8$  种不同的划分。图 2-20(a) 显示了平面上的任意 3 个点  $Z_1, Z_2, Z_3$ 。线性指示函数集对其所作的划分可以表示为

$$[Q(Z_1, \xi), Q(Z_2, \xi), Q(Z_3, \xi)] = [\delta_1, \delta_2, \delta_3]$$

其中各  $\delta_i$  取 0 或 1。这样,最多可以实现 8 种划分。式 (2-93) 给出的函数集是对  $\xi$  决定的超平面两侧进行划分,当  $Z$  位于超平面某一侧时,  $Q(Z, \xi)$  等于 1;在另一侧时,等于 0。当  $N = 2$  此超平面是一条直线。图 2-20(a) 中的 4 条直线正能实现 8 种划分(每条直线实现 2 种划分,它们给出的  $Q(Z, \xi)$  取值正好相反)。下面证明,若  $r = 4$  则对于平面上的任意 4 个点  $Z_1, Z_2, Z_3, Z_4$  无论怎样改变  $\xi$  也不可能实现所有  $2^r = 16$  种不同的划分。图 2-20(b) 给出了一个示例,易于看到,试图用一条直线将平面分成两半的方法把  $Z_1$  和  $Z_3$  划归一

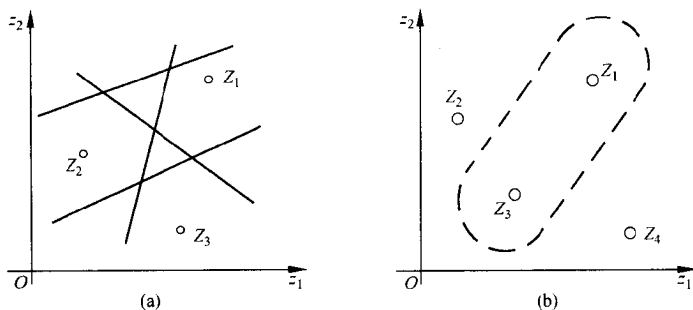


图 2-20 用线性指示函数集作划分的示例

类, 而将  $z_2$  和  $z_4$  划归另一类是行不通的。这就证明了当  $N = 2$  时 此函数集的 VCdim 为  $h = 3$  这符合式 (2-94)。对于  $N > 2$  的任何情况, 只需将上列陈述重复一遍就可以了。

#### 4. VCdim 计算实例之二 —— 线性函数集

$$Q(Z, \xi) = \theta + \sum_{i=1}^N w_i z_i \quad (2-95)$$

其中  $Z$  和  $\xi$  与式 2-93 相同。因为由  $Q(Z, \xi)$  构成的线性指示函数集  $I(Z, \xi, \beta)$  的 VCdim 用式 (2-94) 计算, 所以此函数集的 VCdim 也用该式计算。

#### 5. VCdim 计算实例之三 —— 硬限幅正弦函数集

$$Q(z, \xi) = \text{sgn}[\sin \omega z]$$

其中  $\omega$  和  $z$  都取实数值,  $\xi = \omega$ 。可以证明 此函数集的 VCdim 为  $h = \infty$ 。证明方法如下。对于任何  $r > 0$  设  $z_1 = 10^{-1}, z_2 = 10^{-2}, \dots, z_r = 10^{-r}$ 。  $[Q(z_1, \xi), \dots, Q(z_r, \xi)] = [\delta_1, \dots, \delta_r]$ 。为了实现所有  $2^r$  种划分中的任意一种, 对于某一种划分  $[\delta_1, \dots, \delta_r]$  只需按下式求出  $\omega$  之值即可:

$$\omega = \pi \left[ \sum_{l=1}^r (1 - \delta_l) 10^l + 1 \right]$$

由于  $r$  可取为任意大的正整数, 所以  $h = \infty$ 。

#### 6. VCdim 计算实例之四 —— 隐层取 Sigmoid 函数且输出层取线性函数的 2 层单输出 MLFN

此网络的映射函数由式 (2-82) 描述, 其 VCdim 的准确值迄今未求得。按照文献 [70]、[71] 的推导 若输入为  $N$  维向量  $X$  且隐层含  $N_1$  个神经元 则其 VCdim 由下列不等式界定:

$$4 \left[ \frac{N_1}{2} \right]_1 N \leq h \leq 4 N_w \lg(e N_N) \quad (2-96)$$

其中  $[\cdot]_1$  表示取整数部分,  $N_w$  是网络中权的总数,  $N_N$  是网络中神经元的总数,  $\lg(\cdot)$  表示取以 10 为底的对数。对于取所述参数的网络, 易于求得:  $N_w = [(N+2)N_1 + 1]$ ,  $N_N = N_1 + 1$ 。当  $N$  和  $N_1$  取较大值时 (例如大于 10) 可得,  $[N_1/2]_1 \approx N_1/2$ ,  $[(N+2)N_1 + 1] \approx NN_1$ , 则  $h$  的上、下界近似等于  $4NN_1 \lg[eN_1]$ 、 $2NN_1$ 。若  $N_1$  的取值在 30 ~ 100 范围内 此上界值近似等于  $8NN_1$  这样  $h$  的值可以界定在  $2NN_1 \sim 8NN_1$  范围内。现在通常采用的

一个经验计算公式是  $h = 4NN_1$ 。

## 2.10.4 学习过程推广能力的控制和结构风险最小原理

对于一个学习过程而言，如果已选定了学习机（例如一个 2 层单输出 MLFN）那么可选择的参数只有训练集的规模  $M$  和学习机的规模（例如 2 层单输出 MLFN 的隐层神经元数  $N_1$ ）。后者的作用表现在学习机的  $h$  值。学习机的规模越大，则其函数逼近能力越强，即对于一定的待逼近函数  $f(X)$ ， $R(\xi_0)$  的值越小。但是规模越大， $h$  值也越大。这样，对于一个特定的学习任务可以分成两种情况。第一种情况， $M$  可以不受约束地任意加大。这时可先选择规模足够大的学习机使  $R(\xi_0)$  足够小，然后选择足够大的  $M$  使式 (2-89) 右侧第二、三项足够小，这就能保证  $R(\xi_a)$  足够小，即学习过程有良好的推广能力。第二种情况， $M$  不能任意增大，只能选择为某个可能的最大值。这时学习机的规模成为惟一可选择的参数。

下面以本节讨论的 2 层单输出 MLFN 为例，其规模由隐层神经元数  $N_1$  决定。这时问题归结为在  $M$  一定的条件下如何选择  $N_1$  以使得  $R(\xi_a)$  达到最小值。为此将式 (2-89) 改写为下列形式：

$$R_{N_1}(\xi_a) \leq R_{N_1}(\xi_0) + T_{N_1}(M) \quad (2-97)$$

其中  $R(\xi_a)$  和  $R(\xi_0)$  加下标  $N_1$  是为表明它们随  $N_1$  而变化。根据 Barron 定理， $R_{N_1}(\xi_0)$  的上限由式 (2-86) 决定。 $T_{N_1}(M)$  是式 (2-89) 右侧第 2 项和第 3 项之和，其中的  $\epsilon$  由式 (2-90) 计算且  $h$  可取经验值  $4NN_1$ 。这样，对于一定的  $\gamma$ （例如  $\gamma = 0.01$ ）和  $A, B$  值， $T_{N_1}(M)$  决定于  $M$  和  $N_1$ ，被称为置信限。在  $M$  一定的条件下， $R_{N_1}(\xi_0)$  和  $T_{N_1}(M)$  随  $N_1$  的变化如图 2-21 所示。前者随  $N_1$  的增加而下降，后者相反。

$N_1$  有一最佳值，使  $R_{N_1}(\xi_0)$  与  $T_{N_1}(M)$  之和达到最小，因为它是  $M$  的函数，故记为  $N_1^*(M)$ 。用这种方法来决定网络的规模称为结构风险最小原理。

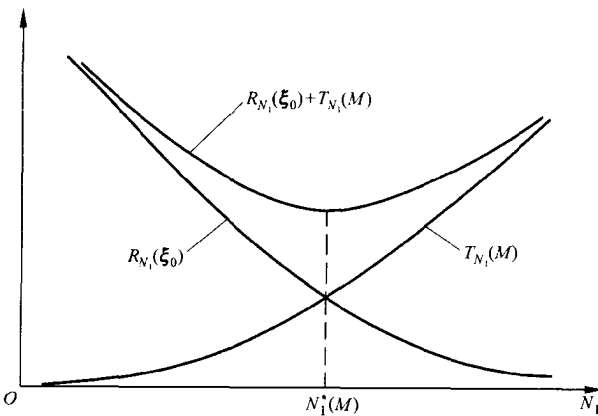


图 2-21  $R_{N_1}$  与  $T_{N_1}$  随  $N_1$  的变化图

## 2.11 提高 MLFN 推广能力的实用方法

统计学习理论给出了训练集规模  $M$  一定的条件下求 2 层单输出 MLFN 最佳隐层神经元个数  $N_1^*(M)$  的方法。这一理论成果与大量经验结果一致,即对于任何训练集规模一定的人工神经网络,网络规模必须选一个恰当的适中值,才能使网络对于训练集内外的数据都具有良好的性能。但是,对于一个具体的学习课题和一个具体的学习机(例如某种类型的 MLFN), $C_f$  和  $h$  等参数都难以准确估算,因此  $N_1^*(M)$  也难以精确计算。更何况究竟 MLFN 的层数如何选择至今仍是一个有争议的问题。因此十分需要一些在实际中易于实现的、确能改善 MLFN 推广能力的算法。已有大量的这一类算法提了出来。下面介绍其中一部分。

### 2.11.1 交叉有效法

将全部训练数据  $(\hat{y}_m, X_m)$  分成两部分。一部分用来对网络进行训练,称为训练集;另一部分对网络进行测试,称为测试集。当网络按照节拍  $k$  进行递推学习时,训练集的风险函数用  $R_{\text{emp}}(\xi(k))$  表示,而测试集的风险函数用  $R_{\text{tes}}(\xi(k))$  表示。随着  $k$  的增加,前者永远呈现下降趋势,而后者先降后升,如图 2-22 所示。这种现象的产生是由于网络规模选得不够恰当(偏大),以致学习越多,则网络参数对训练集内的数据更加适应,而对于未参加训练的数据越不适应。这称为过适应现象。可以看到,如果  $R_{\text{tes}}(\xi(k))$  由降转升的转折点如果是  $k^*$ ,那么学习应该在  $k = k^*$  时终止,否则网络的推广效果将明显转劣。

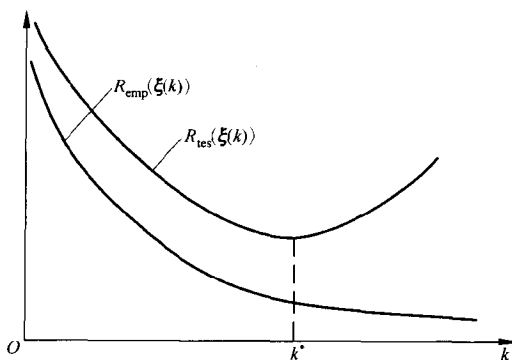


图 2-22 训练集和测试集的风险函数图

### 2.11.2 规格化法

规格化理论和技术(regularization theory and technique)是在解决病态问题(ill-posed problem)时提出来的。20 世纪初 Hadamard 发现,在某些极一般条件下,解下列线性算子方程是病态的:

$$Af = g, \quad f \in \mathcal{F}$$



其中  $f, g$  为  $N$  维和  $Q$  维列向量,  $A$  为  $N \times Q$  维矩阵,  $\mathcal{F}$  是  $f$  的定义域。 $A$  和  $g$  为已知,  $f$  为待求者。所谓病态是指, 即使此式有惟一解, 当  $g$  略有偏差 成为  $g_\delta$  且  $\|g_\delta - g\| = \delta \ll 1$ , 由  $g_\delta$  求得的解  $f_\delta$  与  $f$  之间的差距甚大。即使  $\delta \rightarrow 0$  也不能保证  $f_\delta$  逼近于  $f$ 。可谓差之毫厘, 失之千里。Hadamard 原以为这是一个纯数学问题, 但是后来人们发现许多现实生活中待解决的问题是病态的。其中特别重要的是求因果问题的逆向解, 即由结果求原因。即使是一一对应的因果求逆向解问题, 也是病态的。

1963 年 A. N. Tikhonov 提出用规格化方法解决病态问题<sup>[72,73]</sup>。其思路是 如果求上列线性算子方程时, 是通过求得使下列  $R(f)$  达到最小值的  $f$  来实现的, 则将产生病态问题。

$$R(f) = \|Af - g\|$$

他提出用下列  $\phi(f)$  来替代  $R(f)$ :

$$\phi(f) = \|Af - g\|^2 + \gamma\Omega(f)$$

其中  $\Omega(f)$  是  $f$  的非负、半连续(下端连续)函数。他证明 当  $\Omega(f)$  满足一定条件时, 若  $\gamma \rightarrow 0$  则由  $\phi(f)$  最小求得的解  $f_\gamma$  将收敛于真解  $f$ , 而且可以免除病态问题。 $\gamma\Omega(f)$  即称为规格化项。按照这个思路, 可以用下列带有规格化项的经验风险函数  $\tilde{R}_{\text{emp}}(\xi)$  来替代原经验风险函数  $R_{\text{emp}}(\xi)$  以求得最佳参数  $(\xi_a)$ :

$$R_{\text{emp}}(\xi) = R_{\text{emp}}(\xi) + \gamma\Omega(\xi) \quad (2-98)$$

按照对于规化项  $\gamma\Omega(\xi)$  的不同定义, 可以构成多种改善网络推广性能的学习算法。本节只能择要介绍其中几种。

### 1. 权衰减法

规格化项的定义如下:

$$\gamma\Omega(\xi) = \gamma \sum_{i,j,l} [w_{ij}^{(l)}]^2, \quad \gamma > 0 \quad (2-99)$$

此式右侧和式中包括网络中全部权参数和阈值参数 为简化 二者都用  $w_{ij}$  表示, 按照最陡下降算法, 为求得使  $R_{\text{emp}}(\xi)$  达到最小的  $\xi$  对每一个权  $w_{ij}^{(l)}$  按递推算法进行调整, 每一个递推节拍  $k$  的权调整量  $\Delta w_{ij}^{(l)}(k)$  应采用下列公式计算:

$$\Delta w_{ij}^{(l)}(k) = \left[ -\alpha \frac{\partial R_{\text{emp}}(\xi)}{\partial w_{ij}^{(l)}} - 2\alpha\gamma w_{ij}^{(l)} \right] \Big|_{\xi=\xi(k)} \quad (2-100)$$

已知其中  $\alpha > 0$  为步幅。 $\gamma$  称为规格化系数, 它决定规格化项所起作用的大小。此式右侧第一项的计算已在 2.4 节中给出。第二项的作用可以称为权衰减, 即每调整一步使每个权参数的绝对值减小一些, 减小量取决于  $\gamma, \alpha$  和权本身的幅度(绝对值)。可以将权的减小称为对该权的“惩罚”, 这样, 权本身幅度越大受惩罚越严重。这种方案使网络中各个权的幅度值趋小。实际操作方法是, 随着学习的进展, 如果有某个权的幅度跌落到一个预先设定的很小的门限值以下时, 即可以将其删除。这样就能达到缩小网络规模的目的。

上述方案的缺点是一个权的幅度越大则受惩罚越严重, 这往往导致网络的各个权幅度偏小。为此可以采取下列两种修正方案。第一种方案, 规格化项取下列形式:

$$\gamma\Omega(\xi) = \gamma \sum_{i,j,l} |w_{ij}^{(l)}|, \quad \gamma > 0 \quad (2-101)$$

这时, 式(2-100)右侧第二项将取下列形式:

$$-2\alpha\gamma\text{sgn}[w_{ij}^{(l)}]$$

其中  $\text{sgn}[\cdot]$  为取符号。这个方案的特点是大幅度与小幅度权受到同等的惩罚<sup>[74]</sup> 因而减弱了权幅度趋小的倾向。第二种方案, 规格化项取下列形式:

$$\sum_{i,j,l} \gamma_{ij}^{(l)} [w_{ij}^{(l)}]^2 \quad (2-102)$$

$$\gamma_{ij}^{(l)} = \gamma \{ [w_{ij}^{(l)}]^2 \}^{-p}$$

若  $p=0$  相当于式 (2-99) 若  $p=\frac{1}{2}$  相当于式 (2-101) 的情况 若  $p=\frac{1}{3}$  则是上列二者的中间情况; 若,  $=\frac{2}{3}$  则大幅度的权受惩罚程度轻于小、幅度的权。

## 2. 修正的权衰减法

采取下列的规格化项定义:

$$\gamma\Omega(\xi) = \gamma \sum_{i,j,l} \frac{[w_{ij}^{(l)}/w_0]^2}{1 + [w_{ij}^{(l)}/w_0]^2}, \quad \gamma > 0 \quad (2-103)$$

其中  $w_0$  取某个正值。若  $[w_{ij}^{(l)}/w_0]^2 \gg 1$ , 则上式右侧和式中的对应项接近于 1 这说明该权几乎不受惩罚; 反之, 则对应项接近于  $[w_{ij}^{(l)}/w_0]^2$ 。这样 较  $w_0$  大的权受惩轻而较  $w_0$  小的权受惩重, 从而导致较小幅度的权易于淘汰, 较大幅度的权易于保留。此方案中  $w_0$  要选择适中,  $w_0$  太小则保留的权偏多, 反之则淘汰的权偏多。此外, (1)、(2) 两种权衰减法中  $\gamma$  也必须选择适中 理由与  $w_0$  的情况相似。

## 3. 与隐层神经元输出能量有关的权衰减法

对于一个 2 层单输出 MLFN (当隐层神经元取 Sigmoid 函数时, 网络的映射函数由式 (2-82) 表示) 对于训练集  $(\mathbf{X}_m, \hat{y}_m), m=1 \sim M$  网络的隐层输出为  $x_{mi}^{(1)}, m=1 \sim M, i=1 \sim N_1$ 。这时可以采用下列带有规格化项的经验风险函数<sup>[78]</sup>

$$\tilde{R}_{\text{emp}}(\xi) = R_{\text{emp}}(\xi) + \gamma \left\{ \frac{1}{M} \sum_{m=1}^M \sum_{i=1}^{N_1} \varphi[(x_{mi}^{(1)})^2] \right\} \quad (2-104)$$

其中的函数  $\varphi[\cdot]$  满足下列方程:

$$\frac{d\varphi[u]}{du} = \frac{1}{(1+u)^P}$$

若  $p=1$  则当某个  $x_{mi}^{(1)}$  较大时, 与之相联系的输入至该隐层神经元  $i$  的各个权受惩较轻; 反之 受惩轻重。若  $P>1$  则反差扩大,  $P<1$  则反差缩小。

此外, 还可以将此方案与其他权衰减方案结合起来使用。下面举出两个例子。

$$\tilde{R}_{\text{emp}}(\xi) = \mu R_{\text{emp}}(\xi) + \gamma_1 \left\{ \frac{1}{M} \sum_{m=1}^M \sum_{i=1}^{N_1} \varphi[(x_{mi}^{(1)})^2] \right\} + \gamma_2 \sum_{i,j,l} [w_{ij}^{(l)}]^2 \quad (2-105)$$

$$\tilde{R}_{\text{emp}}(\xi) = \mu R_{\text{emp}}(\xi) + \gamma_1 \left\{ \frac{1}{M} \sum_{m=1}^M \sum_{i=1}^{N_1} \frac{[x_{mi}^{(1)}]^2}{1 + [x_{mi}^{(1)}]^2} \right\} + \gamma_2 \sum_{i,j,l} \frac{[w_{ij}^{(l)}]^2}{1 + [w_{ij}^{(l)}]^2} \quad (2-106)$$

其中  $\mu, \gamma_1, \gamma_2$  皆大于 0。文献[79] 给出了采用式 (2-106) 方案的若干模拟实验结果。当取  $\mu = \gamma_1 = 0.1, \gamma_2 = 0.001$  时 实验证明 无论初始规模如何 隐层神经元数都能达到最佳值。应该说明, 在此算法中, 如果一个隐层神经元的输入权全部被删除了, 则此神经元即应被删除。该神经元在没有输入情况下的固定输出 (等于 0.5) 应乘以相应隐层至输出层的

权值后，并入阈值参数中。

按照规格化法所求得的网络达到了尽可能小的规模（神经元数和权数都尽可能少）。这一结果和前述统计学习理论的理论推导结果一致，也和信息论中的最小描述长度理论（minimum discription length, MDL）一致。

### 2.11.3 删除 (prunning) 法

改善网络推广性能最重要的途径之一是使网络的规模尽可能小。一种简捷的方案是先设计一个规模偏大的网络，然后找到那些“不重要”的权和神经元并将其删除，最终将网络规模缩小到恰当范围。所谓不重要是指某个权或神经元被删除后，经验风险函数  $R_{\text{emp}}(\xi)$  不会起明显变化。这就是说  $R_{\text{emp}}(\xi)$  相对于一个权或一个神经元的敏感度越低，则该权或神经元越应该被删除，所以这种方法也称为敏感度法<sup>[80]</sup>。按照对于敏感度的不同定义，可以构成各种删除算法，下面介绍其中的几种。

#### 1. 关联度 (relevance) 法<sup>[81]</sup>

首先对网络中任一权  $w_{ij}^{(l)}$  的关联度  $\rho_{ij}^{(l)}$  作出定义。 $\rho_{ij}^{(l)}$  等于  $w_{ij}^{(l)}$  删除后与删除前  $R_{\text{emp}}(\xi)$  之差值，即是该权敏感度的度量。由于  $\rho_{ij}^{(l)}$  的直接计算开销太大，所以采用一种间接计算的方法。为此，将通常使用的下列神经元函数

$$x_i^{(l)} = f_l \left[ \sum_{j=1}^{N_{l-1}} w_{ij}^{(l)} x_j^{(l-1)} + \theta_i^{(l)} \right]$$

改写为

$$x_i^{(l)} = f_l \left[ \sum_{j=1}^{N_{l-1}} \zeta_{ij}^{(l)} w_{ij}^{(l)} x_j^{(l-1)} + \zeta_i^{(l)} \theta_i^{(l)} \right] \quad (2-107)$$

这样，可以用下列公式近似计算关联度：

$$\rho_{ij}^{(l)} \approx - \left. \frac{\partial R_{\text{emp}}(\xi)}{\partial \zeta_{ij}^{(l)}} \right|_{\zeta_{ij}^{(l)}=1} \quad (2-108)$$

事实上，上式的计算可以在用 BP 对权进行递推计算的过程中顺便求出；而且因为最终是在  $\zeta_{ij}^{(l)} = 1$  时估计偏微分之处， $\zeta$  参数在最终计算公式中并不出现。

此方法以  $|\rho_{ij}^{(l)}|$  的大小作为  $R_{\text{emp}}(\xi)$  相对于  $w_{ij}^{(l)}$  的敏感度衡量。操作删除的过程是，首先进行正常的 BP 学习，当网络参数收敛到一个最佳值后，删除若干个  $|\rho_{ij}^{(l)}|$  跌落到某个预置门限  $\varepsilon_T$  以下的权，然后再进行 BP 学习，再删除。这一过程反复交替进行，直至网络规模取得了明显的压缩。

还可作两项修正。第一，式 2-108 中的  $R_{\text{emp}}(\xi)$  改用下式的  $\hat{R}_{\text{emp}}(\xi)$ ，

$$\hat{R}_{\text{emp}}(\xi) = \frac{1}{M} \sum_{m=1}^M |\hat{y}_m - f(\mathbf{X}_m, \xi)| \quad (2-109)$$

而  $w_{ij}^{(l)}$  的 BP 递推计算仍采用  $R_{\text{emp}}(\xi)$ 。这一修正使得  $\rho_{ij}^{(l)}$  的估计当  $R_{\text{emp}}(\xi)$  值较小时比原方案更加精确。第二，为了提高关联度估计的准确度，不是在 BP 学习的结尾处对其进行估计，而是在整个 BP 学习过程中对其进行估计，并且引入惯性以取得平均效果。计算公式是

$$\rho_{ij}^{(l)}(k+1) = 0.8 \rho_{ij}^{(l)}(k) + 0.2 \left( - \frac{\partial R_{\text{emp}}(\xi)}{\partial \zeta_{ij}^{(l)}} \right) \bigg|_{\substack{\zeta_{ij}^{(l)}=1 \\ \xi=\xi(k)}} \quad (2-110)$$

一个有趣的现象是根据关联度将网络规模削减后，其容损能力并未降低。即当若干权受损失时，原网络与削减网络的性能相同<sup>[82]</sup>

## 2. 灵敏度法<sup>[83]</sup>

设 BP 递推计算共进行了  $K$  步(从  $k = 0$  开始到  $k = K$  结束)某个权  $w_{ij}^{(l)}$  从初值  $w_{ij}^{(l)}(0)$  始到终值  $w_{ij}^{(l)}(K)$  止。则此权的灵敏度用  $S_{ij}^{(l)}$  表示，它用下列公式计算：

$$S_{ij}^{(l)} = - [R_{\text{emp}}(\xi) \Big|_{\substack{w_{ij}^{(l)}=w_{ij}^{(l)}(K) \\ \text{其他权取恒值}}} - R_{\text{emp}}(\xi) \Big|_{\substack{w_{ij}^{(l)}=w_{ij}^{(l)}(0) \\ \text{其他权取恒值}}}] \frac{w_{ij}^{(l)}(K)}{w_{ij}^{(l)}(K) - w_{ij}^{(l)}(0)} \quad (2-111)$$

此式较难计算，实用中可采用下列近似公式：

$$S_{ij}^{(l)} \approx - \left[ \sum_{k=0}^K \frac{\partial R_{\text{emp}}(\xi)}{\partial w_{ij}^{(l)}} \Big|_{\xi=\xi(k)} \cdot \Delta w_{ij}^{(l)}(k) \right] \frac{w_{ij}^{(l)}(K)}{w_{ij}^{(l)}(K) - w_{ij}^{(l)}(0)} \quad (2-112)$$

如果 BP 递推计算中不含惯性项，由式 (2-29) 得

$$\Delta w_{ij}^{(l)}(k) = -\alpha \frac{\partial R_{\text{emp}}(\xi)}{\partial w_{ij}^{(l)}} \Big|_{\xi=\xi(k)}$$

则式 (2-112) 可表示为

$$S_{ij}^{(l)} = \sum_{k=0}^{K-1} [\Delta w_{ij}^{(l)}(k)]^2 \frac{w_{ij}^{(l)}(K)}{\alpha(w_{ij}^{(l)}(K) - w_{ij}^{(l)}(0))} \quad (2-113)$$

这样，只需记录下递推计算中每一节拍  $k$  的权调整量  $\Delta w_{ij}^{(l)}(k)$  和初始及终结的权值 即能计算  $S_{ij}^{(l)}$ 。 $R_{\text{emp}}(\xi)$  相对于  $w_{ij}^{(l)}$  的敏感度即用  $|S_{ij}^{(l)}|$  的大小来衡量，其小者即可删除。

## 3. 突出特征法<sup>[84]</sup>

假设网络已经过了充分训练，在训练终止后可按下列思路确定  $R_{\text{emp}}(\xi)$  对某个权  $w_{ij}^{(l)}$  的敏感度。设终止时的网络参数为  $\xi(K)$  这时令某个权  $w_{ij}^{(l)}$  变化一个  $w_{ij}^{(l)}$  量而其他权保持不变 则  $R_{\text{emp}}(\xi(K))$  因为此变化量而产生的变化  $R_{\text{emp}}(\xi(K))$  可以用泰勒级数展开来求得：

$$\Delta R_{\text{emp}}(\xi(K)) = \frac{\partial R_{\text{emp}}(\xi)}{\partial w_{ij}^{(l)}} \Big|_{\xi=\xi(K)} \cdot \Delta w_{ij}^{(l)} + \frac{1}{2} \frac{\partial^2 R_{\text{emp}}(\xi)}{\partial [w_{ij}^{(l)}]^2} \Big|_{\xi=\xi(K)} \cdot [\Delta w_{ij}^{(l)}]^2 + \dots$$

泰勒级数中还应包括二阶交叉项、三阶项及其他高次项和交叉项，因其值较小可以略去不计。此外，网络已通过充分学习，所以此式右侧第一项的偏导数相当接近于 0。这样，当  $w_{ij}^{(l)}$  被删除时，使  $\Delta w_{ij}^{(l)} = -w_{ij}^{(l)}(K)$  由之造成的  $\Delta R_{\text{emp}}(\xi(K))$  可用下式近似计算并记之为  $T_{ij}^{(l)}$ 。

$$T_{ij}^{(l)} = \frac{1}{2} \frac{\partial^2 R_{\text{emp}}(\xi)}{\partial [w_{ij}^{(l)}]^2} \Big|_{\xi=\xi(K)} \cdot [w_{ij}^{(l)}(K)]^2 \quad (2-114)$$

$T_{ij}^{(l)}$  称为突出特征，可以作为已训练好的网络对于权  $w_{ij}^{(l)}$  的敏感度。因此可以将  $T_{ij}^{(l)}$  小于某个门限的那些权删除掉。

$T_{ij}^{(l)}$  的求得涉及  $R_{\text{emp}}(\xi)$  相对于  $w_{ij}^{(l)}$  的二阶偏导数的计算。下面以 2 层单输出 MLFN 为例说明其计算方法（见式 (2-82)）。首先，对于训练集中的每一组数据  $(\mathbf{X}_m, \hat{y}_m)$ ， $m = 1 \sim M$ ，通过前向计算求得网络中各神经元的输出。

隐层各神经元的输出：

$$x_{mi}^{(1)} = f_s[I_{mi}^{(1)}], \quad I_{mi}^{(1)} = \sum_{j=1}^N w_{ij}^{(1)} x_{mj} + \theta_i^{(1)}, \quad i = 1 \sim N_1$$

输出层单神经元的输出：

$$x_{m1}^{(2)} = y_m = \sum_{i=1}^{N_1} w_{1i}^{(2)} x_{mi}^{(1)} + \theta_1^{(2)}$$

引用式(2-11)和式(2-12)得到

$$R_{\text{emp}}(\xi) = \frac{1}{M} \sum_{m=1}^M (\hat{y}_m - y_m)^2 = \frac{1}{M} \sum_{m=1}^M (\hat{y}_m - x_{m1}^{(2)})^2$$

通过反向计算，即可求得下列二阶偏微分：

$$\frac{\partial^2 R_{\text{emp}}(\xi)}{\partial [w_{1i}^{(2)}]^2} = \frac{2}{M} \sum_{m=1}^M [x_{mi}^{(1)}]^2, \quad i = 1 \sim N_1$$

$$\frac{\partial^2 R_{\text{emp}}(\xi)}{\partial [w_{ji}^{(1)}]^2} = \frac{2}{M} \sum_{m=1}^M \{ [f'_s(I_{mi}^{(1)})]^2 (w_{1i}^{(2)})^2 - (\hat{y}_m - y_m) f''_s(I_{mi}^{(1)}) w_{1i}^{(2)} \} (x_{mj}^{(1)})^2,$$

$$i = 1 \sim N_1, \quad j = 1 \sim N$$

$$f'_s(I_{mi}^{(1)}) = x_{mi}^{(1)} (1 - x_{mi}^{(1)})$$

$$f''_s(I_{mi}^{(1)}) = (1 - 2x_{mi}^{(1)}) x_{mi}^{(1)} (1 - x_{mi}^{(1)})$$

#### 4. 交叉删除法<sup>[85]</sup>

对于一个2层单输出 MLFN(式2-82))，其隐层各神经元的功能取决于它的各输入权构成的权向量  $\mathbf{W}_i^{(1)}$ ， $\mathbf{W}_i^{(1)} = [w_{i1}^{(1)}, \dots, w_{iN}^{(1)}]$ ， $i = 1 \sim N_1$ 。如果有两个权向量，它们之间的“夹角”很小，这表明二者的功能近似，其中的一个作为多余者即可予以删除。具体做法是首先将隐层各权向量写成下列形式：

$$\mathbf{W}_i^{(1)} = G_i^{(1)} \hat{\mathbf{W}}_i^{(1)}, \quad i = 1 \sim N$$

其中  $\|\hat{\mathbf{W}}_i^{(1)}\| \equiv 1$ ， $G_i^{(1)} = \|\mathbf{W}_i^{(1)}\|$ ， $G_i^{(1)}$  称为  $\mathbf{W}_i^{(1)}$  的增益或幅度。现在，在 BP 递推算法的每一节拍  $k$ ，在进行了各个权的调整后再增加一个增益调整操作，每个增益的调整量  $\Delta G_i^{(1)}$  可以用下式计算：

$$\Delta G_i^{(1)} = -\mu \sum_{\substack{j=1 \\ j \neq i}}^{N_1} (\hat{\mathbf{W}}_i^{(1)} \cdot \hat{\mathbf{W}}_j^{(1)})^2 G_j^{(1)}, \quad i = 1 \sim N_1 \quad (2-115)$$

其中  $\hat{\mathbf{W}}_i^{(1)} \cdot \hat{\mathbf{W}}_j^{(1)}$  即等于  $\mathbf{W}_i^{(1)}$  和  $\mathbf{W}_j^{(1)}$  之间夹角的余弦，当二个向量越相似，此值越接近于1。 $\mu > 0$ ，称为增益调整步幅。实际操作时，可以进行若干步权调整再做一次增益调整。当某个增益  $G_i^{(1)}$  低于一个小门限值  $\epsilon$  ( $\epsilon > 0$ ) 或小于0时，隐层第  $i$  神经元即可予以删除。 $\mu$  值应恰当选择， $\mu$  值太小时不起删除作用，反之则删除过多。

#### 2.11.4 局部连接和权分享<sup>[86,87]</sup>

对于图像、符号、文字、语音等进行模式识别时，输入空域或时域信号中的某个特征往往只出现在一个小局部区域中，而此局部区域又可能位于全域的任何一个位置。这样，为了检测某个局部特征，MLFN 第一隐层的各个神经元不必与表示全部特征的输入向量  $\mathbf{X}$  的每个分量相连接，而只需与表示局部特征的若干个分量相连接。例如，一个分辨率为  $16 \times 16$  的图像中包含 256 个像素，它们构成了一个 256 维向量作为 MLFN 的输入。若某些局部特征只表现在一个  $5 \times 5$  的局部区域中，那么网络第一隐层的某个为发现此局部特

征的神经元只需与  $\mathbf{X}$  的 256 个分量中的 25 个分量相连接，从而能够使权的数量有很大压缩。此外，从第一隐层向更上隐层传送信号时也可以采取类似的原理。图 2-23 给出了一个 2 层单输出局部连接网络的示例，其中输入是一个 4 维向量  $\mathbf{X}$ ,  $\mathbf{X} = [x_1, x_2, x_3, x_4]$ 。输入层还包含一个为提供阈值参数而设的单元（输出为 1）。隐层含 6 个神经元和 1 个输出恒为 1 的单元（提供阈值参数）。如果采取全连接方案，网络中权的总数是 37。而按照局部连接方案，隐层神经元分成  $A, B, C$  和  $(D, E, F)$  两组，它们分别用来检测两种不同的局部特征。 $A$  和  $D$  只与  $x_1, x_2$  相连， $B$  和  $E$  只与  $x_2, x_3$  相连， $C$  和  $F$  只与  $x_3, x_4$  相连。 $A, B, C$  取相同输入权值， $D, E, F$  取相同输入权值，这称为权分享。这样，权的总数降至 25，而可调权参数只有 13 个（因为若干权用同一参数）。在解决一些实际问题（如字符、语音、图像的识别）时网络规模很大，采用此方案可有效压缩权的个数从而改善其推广性能。

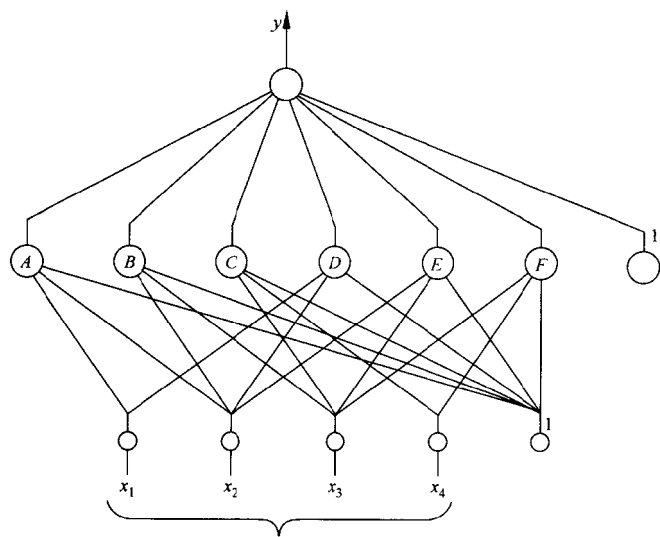


图 2-23 2 层单输出局部连接网络示意图

### 2. 11. 5 局部优化法

一个待逼近函数  $y = \hat{f}(\mathbf{X})$  在  $\mathbf{X}$  的全部定义域中的变化如果非常复杂，而将定义域分割为若干个小局部区域后，每个小区内的函数变化就简单多了。这时每个小区可以用一单独的网络，其规模较小，因而推广性能较好。图 2-24 给出了一个示例  $y = f(x)$ 。如果将  $x$  的全部定义域  $[x_1, x_5]$  分成 4 个小区，每个区中函数变化相对简单；例如， $[x_4, x_5]$  这个小区中的函数可以用一直线来逼近。小区划分得越多则每个小区中的函数变化越简单，因而网络也越简单。但是小区划分得越多使计算开销越大；此外，当训练集规模一定时，小区越多则每个小区分配到的训练样本越少，这有可能使每个小区网络的推广性能反而劣化。因此，必须对小区的个数作折衷选择。

划分小区的一种简单方法是将  $\mathbf{X}$  的每一维的定义区间（例如通过规格化后规整在  $[-1, 1]$  区间内）划分为  $Q$  个等间隔区。若  $\mathbf{X}$  为  $N$  维，则共划分为  $Q^N$  个小区。这一方案的

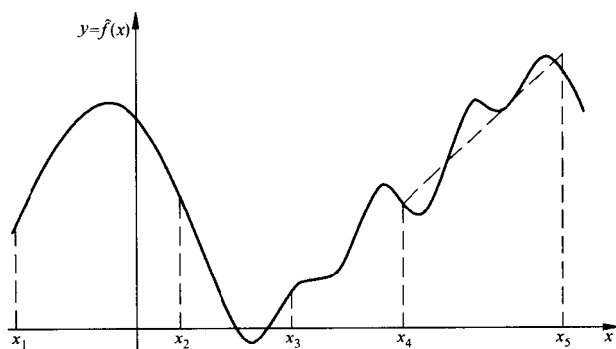


图 2-24  $y = f(x)$  的示意图

明显缺陷是当  $N$  较大时 小区数将大得惊人(例如 当  $N = 10$  即使  $Q = 2$  也将有 1024 个小区)。一种实用的方案是用向量量化的方法将  $\mathbf{X}$  分为若干聚类区, 每个区用一个网络, 这时小区的数字很容易得到控制。

## 2.11.6 加噪声法

如果在训练时, 在训练样本  $\mathbf{X}_m$  和  $\hat{y}_m$  上叠加小噪声, 实验结果证明, 网络的推广性能将得到改善<sup>[88]</sup>

## 2.11.7 多网络法<sup>[89~95,167]</sup>

### 1. 模式识别情况

可以用  $2Q + 1$  个网络代替单个网络来完成识别任务, 其中  $Q$  是一个正整数。这  $2Q + 1$  个网络应具有不同的权参数, 在识别时可能给出互异的识别结果, 最后用少数服从多数的表决法来决定识别输出。为了获得  $2Q + 1$  个网络 可以采用下列方案。第一 各网络的结构和规模相同且训练集相同, 只是权初值不同。这样, 在学习结束时造成网络具有不同权值, 它们相应于  $R_{\text{emp}}(\xi)$  的不同局部极小点。文献 [89] 按此方案所做的实验结果表明, 各网络给出的识别结果确有差异, 采用表决法确可改善网络的集外正确识别率, 即提高了推广性能。第二, 将训练集分成  $2Q + 1$  个子集, 每个网络用不同的训练集进行训练, 从而导致不同的权值。这一方案只有当训练集规模较大时才适用。第三, 令各个网络的结构不同。例如 隐层数不同 隐层神经元数不同 甚至隐层神经元函数不同。第四 以上几种方法的组合。

### 2. 函数逼近情况

可以采用  $P$  个网络实现同一映射  $y = f(\mathbf{X})$  并且用 1 所列的各种方案使各个网络有所不同。在应用时 取  $P$  个网络输出的算术平均值作为所需输出。文献 [92] 证明, 这种方案可以使得按均方误差准则衡量的推广性能得到改善。

### 3. 多网络法与交叉有效法相结合

将全部训练数据分成训练集和测试集两部分。当各个网络用训练集进行学习后, 用测

试集予以检验，分出其优劣。在应用时，按各网络的测试劣优度，取其输出的加权和作为所需输出（优者加权大，劣者反之）

#### 4. 用多个 MISO 网络替代单个 MIMO 网络

如果待实现的映射输出  $\mathbf{Y}$  是一维数大于 1 的向量（设为  $N_L$  维）此映射一般用单个网络来实现。但是，这也可以用  $N_L$  个 MISO 网络来实现，每个网络的输出等于  $\mathbf{Y}$  的一个分量。这一方案有可能使每一分网络比较简单，从而改善推广性能。在模式识别领域中，如果  $\mathbf{X}$  属于  $\mathcal{L}$  种类别，则  $N_L$  应等于  $\mathcal{L}$ 。当  $\mathbf{X}$  为第  $i$  类时，网络第  $i$  端输出应等于 1 而其他各端应等于 0。用多个单输出网络时，对相应于第  $i$  类的网络，当  $\mathbf{X}$  为第  $i$  类时该输出应等于 1，而为其他各类时，输出应等于 0。

## 2.12 MLFN 作为后验概率估值器

设输入向量  $\mathbf{X}$  与  $\mathcal{L}$  个类别  $C^{(l)}, l = 1 \sim \mathcal{L}$ ，相联系。它们之间的关系用联合概率分布函数  $P(\mathbf{X}, C^{(l)})$  描述。 $P(\mathbf{X}, C^{(l)}) = P(\mathbf{X})P(C^{(l)}|\mathbf{X})$ 。正确估计后验概率  $P(C^{(l)}|\mathbf{X})$  是一个有很多实际用途但又是一个难以解决的课题。用传统的统计方法进行估计时，需要对  $P(C^{(l)}|\mathbf{X})$  的形式做某种假设（例如假设其为高斯函数、若干个高斯函数相加，等等）。这种假设往往与实际情况不相符。此外，为了使计算简化，常需对计算过程作一些简化。这使得传统方法的估计精度不高。此外，传统方法是基于对假设的概率分布函数的参数进行估计，而参数估计本身是一个病态问题<sup>[69]</sup>。MLFN 可以实现任意形式的复杂映射关系，其参数是根据 ERM 归纳原理求得的，这使得它实现的函数估计准确度高于传统方法。它的其他很多优于传统方法之处将在 2.16 节中进一步陈述。

下面将证明，如果用 MLFN 来实现一个分类器且采取均方误差准则作为构成经验风险函数的标准对网络进行训练，那么该网络也能成为一个后验概率估值器。设有  $\mathcal{L}$  种类别，记为  $\{C^{(1)}, \dots, C^{(\mathcal{L})}\}$ 。现有一个规模为  $M$  的训练集  $\{(\mathbf{X}_m, C_m), m = 1 \sim M, C_m \text{ 表示 } \mathbf{X}_m \text{ 所属的类别}, C_m \in \{C^{(1)}, \dots, C^{(\mathcal{L})}\}\}$ 。若采用一个 MLFN，其输入为  $\mathbf{x}$ ，输出为  $\mathcal{L}$  维向量  $\mathbf{Y}$ ， $\mathbf{Y} = [y_1, \dots, y_{\mathcal{L}}]$ 。在对网络进行训练时，若取  $\mathbf{X}_m$  为网络输入，网络的理想输出记为  $\hat{\mathbf{Y}}_m = [\hat{y}_{m1}, \dots, \hat{y}_{m\mathcal{L}}]$ 。当  $C_m = C^{(l)}$ ，则  $\hat{y}_{mi} = \delta_{il}, i = 1 \sim \mathcal{L}; \delta_{il} = 1, i = l; \delta_{il} = 0, i \neq l$ 。训练完成后，网络作为分类器用时，每输入一个  $\mathbf{x}$  即可以把网络的  $\mathcal{L}$  个输出中最大输出端的序号作为  $\mathbf{x}$  所属类别的编号。网络作为后验概率估值器时，每个输出  $y_i$  应等于  $P(C^{(i)}|\mathbf{X})$ 。

若网络的  $\mathcal{L}$  个输出表示为  $y_i(\mathbf{X}, \boldsymbol{\xi}), i = 1 \sim \mathcal{L}$ ，则基于均方准则的经验风险函数  $R_{\text{emp}}(\boldsymbol{\xi})$  可以表示为

$$R_{\text{emp}}(\boldsymbol{\xi}) = \frac{1}{M} \sum_{m=1}^M \sum_{i=1}^{\mathcal{L}} [\hat{y}_{mi} - y_i(\mathbf{X}, \boldsymbol{\xi})]^2 \quad (2-116)$$

另一种是相对熵函数形式的经验风险函数，为

$$R_{\text{emp}}(\boldsymbol{\xi}) = \frac{1}{M} \sum_{m=1}^M \sum_{i=1}^{\mathcal{L}} \hat{y}_{mi} \ln \frac{\hat{y}_{mi}}{y_i(\mathbf{X}, \boldsymbol{\xi})} \quad (2-117)$$

无论基于哪一种形式，都可以证明，只要网络中的权足够多，如能求得最佳参数  $\boldsymbol{\xi}_a$ ，使得  $R_{\text{emp}}(\boldsymbol{\xi}_a)$  为全局最小值，那么以  $\boldsymbol{\xi}_a$  为参数的网络能够实现所需的后验概率估计。下面仅就



第一种情况予以证明。

首先, 对于训练集中任一  $\mathbf{X}_m$  成立

$$\sum_{l=1}^{\mathcal{L}} P(C^{(l)} / \mathbf{X}_m) = 1 \quad (2-118)$$

其次 将式 (2-116) 表示为下列形式:

$$R_{\text{emp}}(\xi) = \frac{1}{M} \sum_{m=1}^M \sum_{l=1}^{\mathcal{L}} \sum_{i=1}^{\mathcal{L}} [\hat{y}_{mi} - y_i(\mathbf{X}_m, \xi)]^2 P(C^{(l)} / \mathbf{X}_m)$$

注意, 此式可改写为

$$R_{\text{emp}}(\xi) = \frac{1}{M} \sum_{m=1}^M \left\{ \sum_{i=1}^{\mathcal{L}} [\hat{y}_{mi} - y_i(\mathbf{X}_m, \xi)]^2 P(C^{(i)} / \mathbf{X}_m) + \sum_{i=1}^{\mathcal{L}} [\hat{y}_{mi} - y_i(\mathbf{X}_m, \xi)]^2 \left[ \sum_{\substack{l=1 \\ l \neq i}}^{\mathcal{L}} P(C^{(l)} / \mathbf{X}_m) \right] \right\}$$

当  $\mathbf{X}_m \in C^{(i)}$  时 应有  $\hat{y}_{mi} = 1$  当  $\mathbf{X}_m \notin C^{(i)}$  时 应有  $\hat{y}_{mi} = 0$  且有

$$\sum_{\substack{l=1 \\ l \neq i}}^{\mathcal{L}} P(C^{(l)} / \mathbf{X}_m) = 1 - P(C^{(i)} / \mathbf{X}_m)$$

这样 前式中  $\{\cdot\}$  内项可表示为

$$\sum_{i=1}^{\mathcal{L}} [1 - y_i(\mathbf{X}_m, \xi)]^2 P(C^{(i)} / \mathbf{X}_m) + \sum_{i=1}^{\mathcal{L}} [0 - y_i(\mathbf{X}_m, \xi)]^2 (1 - P(C^{(i)} / \mathbf{X}_m))$$

作简单推导, 即可得

$$R_{\text{emp}}(\xi) = \frac{1}{M} \sum_{m=1}^M \sum_{i=1}^{\mathcal{L}} [y_i^2(\mathbf{X}_m, \xi) - 2y_i(\mathbf{X}_m, \xi)P(C^{(i)} / \mathbf{X}_m) + P^2(C^{(i)} / \mathbf{X}_m)] + \frac{1}{M} \sum_{m=1}^M \sum_{i=1}^{\mathcal{L}} [P(C^{(i)} / \mathbf{X}_m) - P^2(C^{(i)} / \mathbf{X}_m)]$$

此式右侧第二个和式与  $\xi$  无关, 所以只需求得某个  $\xi_a$ , 使该式右侧第一个和式达到全局最小值 就能使  $R_{\text{emp}}(\xi)$  达到全局最小值。而只有当下列条件成立时该式达到全局最小,

$$y_i(\mathbf{X}_m, \xi_a) = P(C^{(i)} / \mathbf{X}_m) \quad i = 1 \sim \mathcal{L}$$

如果一个网络的规模足够大, 从而能提供满足上列条件的权参数  $\xi_a$ 。则当网络输入为  $\mathbf{X}_m$  时 网络的第  $i$  端输出值即等于  $P(C^{(i)} / \mathbf{X}_m)$ 。另外 当训练集的规模足够大时 这一结论还可以推广到  $\mathbf{X}$  定义域中所有的向量。

## 2.13 递归神经网络

### 2.13.1 两个离散时间序列之间的映射

本章前述各节所讨论的 MLFN 都是为了实现单个输入向量  $\mathbf{X}$  至单个输出向量  $\mathbf{Y}$  的映射, 这可以称为静态映射。而在非线性动力系统的建模、辨识、控制、故障诊断以及时间序列预测等许多领域中均涉及两个离散时间序列  $\mathbf{X}(\nu)$  和  $\mathbf{Y}(\nu)$  之间的映射, 它们分别

对应于一个非线性动力系统的输入和输出， $\nu$  是离散时间变量。这时，某个离散时刻  $\nu$  的输出  $Y(\nu)$  不仅依赖于  $X(\nu)$ ，而且还可能依赖于  $X(\nu-1), X(\nu-2), \dots$  以及  $Y(\nu-1), Y(\nu-2), \dots$ ，这可以称之为一种动态映射。为了用 MLFN 来实现动态映射，下面分几种情况来讨论。

(1) 如果对于任一离散时刻  $\nu, Y(\nu)$  只取决于  $X(\nu)$ ，则与静态映射的情况并无差异。这时网络的设计与学习算法与前述各节相同。

(2) 如果对于任一离散时刻  $\nu, Y(\nu)$  取决于  $X(\nu), X(\nu-1), \dots, X(\nu-P)$ ，而与  $Y(\nu-1), Y(\nu-2), \dots$  无关。这时可以用  $P$  个延时——存储单元将  $X(\nu)$  之前的  $P$  个输入存储起来。网络运行时以  $X(\nu), X(\nu-1), \dots, X(\nu-P)$  作为输入，以  $Y(\nu)$  作为输出。这种网络实现的仍是一种静态映射，只是将输入范围扩大而已。一般将其称为时延神经网络 (TDNN)。在数字信号处理和系统理论等学科中，与这种网络对应的线性系统称为 FIR 或 MA 模型。而现在用一个非线性的 MLFN 来替代线性系统如图 2-25 所示。可以称之为 NFIR 或 NMA 模型 (N 表示非线性)，其设计与学习算法可依照前述各节的方案。如给定训练集  $(X(\nu), Y(\nu)), \nu = 1, 2, \dots$ ，则要求网络经过训练以后，当网络的输入为  $X(\nu)$  时，网络的实际输出  $Y(\nu)$  与理想输出  $Y(\nu)$  之间的均方差达到最小。

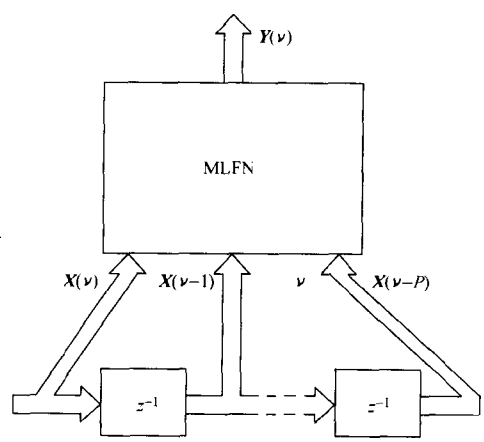


图 2-25 NFIR 或 NMA 模型

(3) 如果  $Y(\nu)$  不仅取决于  $X(\nu), X(\nu-1), \dots, X(\nu-P)$ ，而且取决于  $Y(\nu-1), Y(\nu-2), \dots, Y(\nu-Q)$  这时可以用图 2-26 的网络结构对一个 MLFN 进行训练。网络的输入  $X(\nu), X(\nu-1), \dots, X(\nu-P)$  和  $\hat{Y}(\nu-1), \hat{Y}(\nu-2), \dots, \hat{Y}(\nu-Q)$  取自训练集  $(X(\nu), Y(\nu)), \nu = 1, 2, \dots$  网络的实际输出  $Y(\nu)$  通过训练后与  $\hat{Y}(\nu)$  之间的均方差为最小。这种映射仍是静态的，所以可采取前述的各种学习算法。

网络的训练完成后，可以按照两种方式投入实际运行。第一种方式，不但在离散时刻  $\nu$  及  $\nu$  以前诸时刻的非线性动力系统输入  $X(\nu), X(\nu-1), \dots$  是已知的，而且  $\nu$  以前的诸系

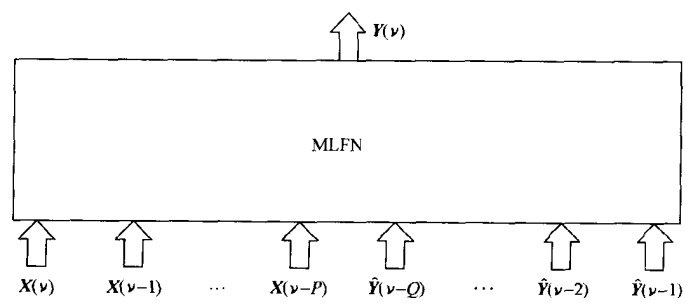


图 2-26 对 MLFN 进行训练的网络结构图

统输出也是已知的，可表示为  $Y(\nu-1), Y(\nu-2), \dots$  (加“ $\hat{\cdot}$ ”标记是表明其为系统的实际输出)。这时可以按照图 2-27 所示的结构来建立一个非线性动力系统模型，其中 MLFN 的结构和参数已在训练中确定，网络输出  $\hat{Y}(\nu)$  在最小均方意义上逼近被建模的实际系统真实输出  $Y(\nu)$ 。第二种方式， $\nu$  以前的诸系统输出不可得知，这时不可能再按图 2-27 的结构对非线性系统建模，而只能取图 2-28 所示的结构。后者的  $Y(\nu-1), Y(\nu-2), \dots$  是由

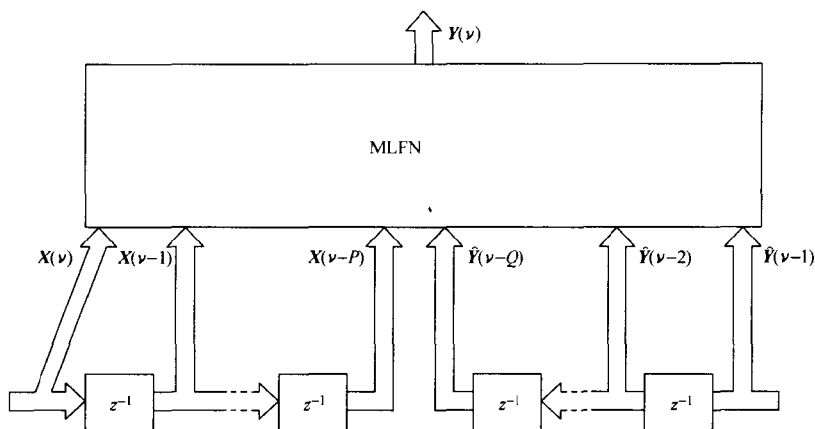


图 2-27 第一种方式所用的网络结构

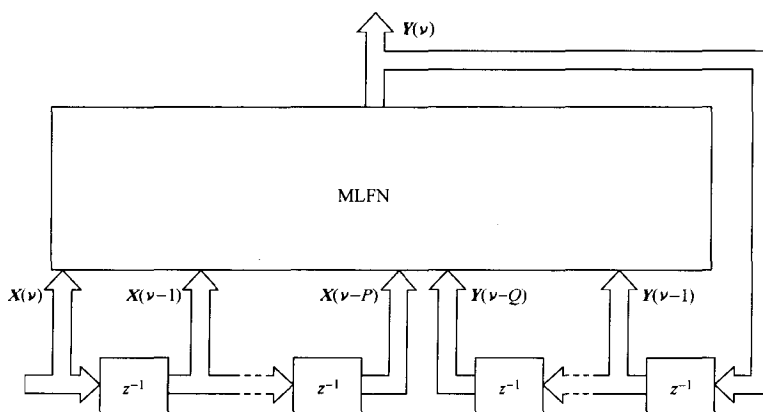


图 2-28 第二种方式所用的网络结构

MLFN 的输出反馈回来，以取代实际系统的真实输出  $\hat{Y}(\nu-1), \hat{Y}(\nu-2), \dots$ 。第一种方式实现的是一种静态映射结构，常称为单步预测方式。第二种方式实现的是一种动态（有反馈的或递归的）映射结构，常称为多步预测方式。在许多实际应用中必须按第二种方式工作。但这种方式存在如下缺陷：首先，网络的输出  $\hat{Y}(\nu)$  与系统真实输出  $Y(\nu)$  不一致，将其反馈后将引起误差。随着  $\nu$  的增加，误差逐渐积累，使得网络输出与真实输出之间的差异逐渐扩大。这样，这种方式只能适用于一短段时间中。其次，这种方式虽然具有动态（反馈）结构，但其训练却是静态的，二者不能协调。第三，当  $P$  和  $Q$  较大时，需要很多延时存储环节，这不但使存

储开销加大而且使 MLFN 的输入维数非常大，这使网络的学习效率很低。因此，更有效的方案是采取一种具有简单递归结构且训练与运行一致的结构，这就是下面讨论的 RNN。

2.13.2 内反馈 RNN

RNN 是 recurrent neural network 的缩写。一种典型的内反馈 RNN 的结构如图 2-29 所示，此网络由  $L$  层构成；输出层含  $N_L$  个神经元，皆取线性函数，第  $l$  隐层含  $N_l$  个神经元 ( $l = 1 \sim L - 1$ ) 皆取 Sigmoid 函数。输入层 ( $l = 0$ ) 含  $N_0$  个神经元，与网络输入向量的各维分量对应。网络按离散时间  $\nu$  运行，在时刻  $\nu$  的网络各神经元输出记为  $x_i^{(l)}(\nu), i = 1 \sim N_l, l = 1 \sim L$ 。  $x_i^{(0)}(\nu) = x_i(\nu), i = 1 \sim N_0$  是输入信号且  $N_0 = N$  是输入向量的维数。  $x_i^{(L)}(\nu) = y_i(\nu), i = 1 \sim N_L$  是输出信号；  $N_L$  是输出向量维数。网络中除了普通 MLFN 中由输入至输出的信号前向传送途径外，每个隐层的各神经元之间存在全反馈连接。图 2-29 中显示了  $1 \sim L - 1$  各隐层中的一个隐层 (第  $l$  层)，它的各神经元输出  $x_1^{(l)}(\nu), \dots, x_{N_l}^{(l)}(\nu)$  各延时一个节拍后，得到  $x_1^{(l)}(\nu - 1), \dots, x_{N_l}^{(l)}(\nu - 1)$ ，再反馈至同一层的各神经元，与下层上传的信号结合为各神经元的输入信号。注意，反馈只存在于每一隐层内，输出层无反馈。此外，也可以构成具有多个节拍的延时反馈网络，但是这种结构既增加了复杂性又无助于提高网络性能，所以下面将只讨论具有一个节拍延时反馈的 RNN。RNN 与数字信号处理中的 IIR 相对应，也与线性系统中的 ARMA 模型对应。它是一有反馈的非线性动力系统，可实现输入至输出的非线性动态映射，因此可以称为 NIIR 滤波器，或称为 NARMA 模型。众所周知，基于线性 AR 模型的线性预测编码 (LPC) 算法是一种高效的建模算法，其局限性在于模型中不含反馈且不含非线性。但是线性 ARMA 模型的建模计算很困难，NARMA 模型就更困难了。RNN 提供了一个实现 NARMA 模型的途径，其重要意义是不言而喻的。

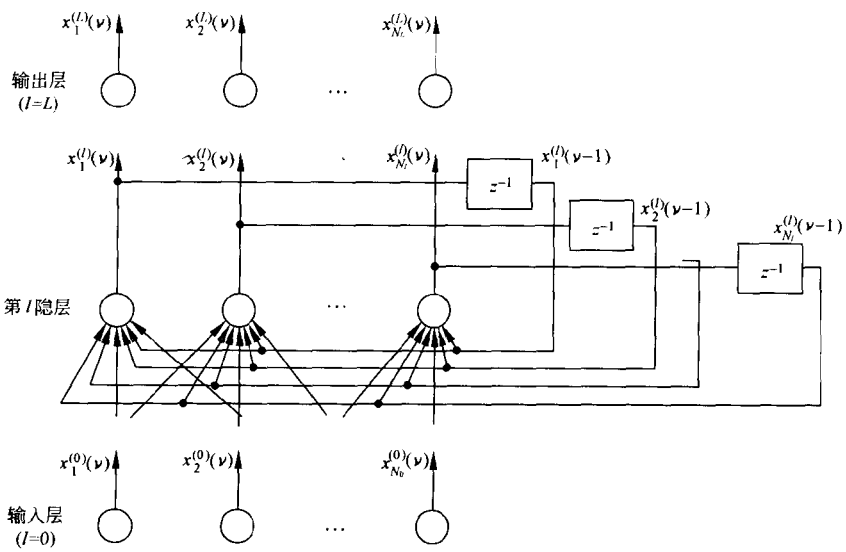


图 2-29 内反馈 RNN 结构图

对于 RNN, 网络的训练集是两个相互对应的离散时间序列  $\{\mathbf{X}(\nu), \hat{\mathbf{Y}}(\nu)\}, \nu = 1, 2, \dots$ 。在对网络进行训练时, 令输入向量序列为  $\mathbf{X}^{(0)}(\nu) = \mathbf{X}(\nu)$ , 令理想输出向量序列为  $\mathbf{X}^{(L)}(\nu) = \mathbf{Y}(\nu)$ , 而网络的实际输出向量序列为  $\mathbf{X}^{(L)}(\nu)$ 。网络经过训练后, 应使得  $\mathbf{X}^{(L)}(\nu)$  与  $\mathbf{X}^{(L)}(\nu)$  之间的均方误差达到最小。

网络的运行过程如下。首先, 按照离散时间  $\nu = 1, 2, \dots$  向网络输入向量  $\mathbf{X}(\nu)$ , 令  $\mathbf{X}^{(0)}(\nu) = \mathbf{X}(\nu), \nu \geq 1$ ; 而且在网络开始运行时, 设定初值  $\mathbf{X}_i^{(l)}(\nu) = 0, \forall i = 1 \sim N_l, l = 1 \sim L-1, \nu = 0$ 。这样对于  $\nu > 0$ , 可以按下列公式计算各  $x_i^{(l)}(\nu)$

$$\left. \begin{aligned} x_i^{(l)}(\nu) &= f_l[I_i^{(l)}(\nu)], l = 1 \sim L, i = 1 \sim N_l, \nu = 1, 2, \dots \\ I_i^{(l)}(\nu) &= \sum_{j=1}^{N_l} w_{ij}^{(l,l)} x_j^{(l)}(\nu-1) + \sum_{j=1}^{N_{l-1}} w_{ij}^{(l,l-1)} x_j^{(l-1)}(\nu) + \theta_i^{(l)}, l = 1 \sim L-1 \\ I_i^{(L)}(\nu) &= \sum_{j=1}^{N_{L-1}} w_{ij}^{(L,L-1)} x_j^{(L-1)}(\nu) + \theta_i^{(L)}, l = L \end{aligned} \right\} \quad (2-119)$$

当  $l = 1 \sim L-1, f_l[\cdot]$  为 Sigmoid 函数,  $I_i^{(l)}(\nu)$  右侧第一个和式是第  $l$  层各神经元的输出延迟一拍后对本层第  $i$  个神经元的反馈输入, 第二个和式是第  $l-1$  层各神经元至第  $l$  层第  $i$  神经元的前向输入,  $\theta_i^{(l)}$  是阈值。当  $l = L, f_l[\cdot]$  为线性函数,  $I_i^{(L)}(\nu)$  仅由下层传送来的前向输入和阈值构成而不含本层反馈。

如果给定训练集  $\{\mathbf{X}^{(0)}(\nu), \mathbf{X}^{(L)}(\nu)\}, \nu = 1, 2, \dots$ , 而网络的实际输出为  $\mathbf{X}^{(L)}(\nu)$ , 则通过训练后应使得  $\mathbf{X}^{(L)}(\nu)$  与  $\mathbf{X}^{(L)}(\nu)$  之间的误差在某种定义下达到最小。下面将介绍目前常用的两种训练算法, 其中一种是动态梯度算法, 另一种是扩展卡尔曼滤波器算法 (extended Kalman filter - EKF)。

### 2.13.3 动态梯度学习算法 [140,143,157]

设网络中全部权 (包括阈值) 构成向量  $\xi, \mathbf{X}^{(L)}(\nu)$  是  $\xi$  的函数 (为简洁起见, 并未特地将其标明)。动态梯度算法的目标是调整  $\xi$  使得  $\mathbf{X}^{(L)}(\nu)$  与  $\hat{\mathbf{X}}^{(L)}(\nu)$  之间的均方误差达到最小。为此, 首先定义时刻  $\nu$  的瞬时风险函数  $R_{\text{emp}}^{(\nu)}(\xi)$  如下:

$$R_{\text{emp}}^{(\nu)}(\xi) = \|\mathbf{X}^{(L)}(\nu) - \hat{\mathbf{X}}^{(L)}(\nu)\|^2 = \sum_{i=1}^{N_L} [\hat{x}_i^{(L)}(\nu) - x_i^{(L)}(\nu)]^2 \quad (2-120)$$

训练可以按照离线批处理方式或在线自适应方式进行。按批处理方式时, 首先应采集到一个长度足够的训练集, 设其取值范围为  $\nu = 1, 2, \dots, \mathcal{N}$  ( $\mathcal{N}$  足够大)。这时可以定义一个针对此训练集的风险函数,

$$R_{\text{emp}}(\xi) = \sum_{\nu=1}^{\mathcal{N}} R_{\text{emp}}^{(\nu)}(\xi) \quad (2-121)$$

学习目的是求最佳参数向量  $\xi_a$ , 使得上述风险函数达到极小值。按照在线 (实时) 自适应方式时, 可定义下列随  $\nu$  而改变的短时风险函数:

$$R_{\text{emp}}(\xi, \nu) = \sum_{\mu=\nu-\eta}^{\nu-1} R_{\text{emp}}^{(\mu)}(\xi) \quad (2-122)$$

这时可随时采集训练数据，随时对网络进行训练，学习目标是求随  $\nu$  而变化的最佳参数向量  $\xi_a(\nu)$ ，使得上述短时风险函数达到最小。取和间隔  $\eta$  越长，则可取得更好的取平均效果而自适应能力减弱，反之则自适应效果改善。由于上述两种训练方式的参数学习算法并无本质区别，下面仅就离线批处理方式予以讨论。

在训练开始前，首先对网络中所有的权设定随机初值  $w_{ij}^{(l,D)}(0), w_{ij}^{(l,l-1)}(0), \forall l, i, j$ 。然后按照最陡下降算法，以节拍  $k$  进行下列递推计算：

$$\left. \begin{aligned} w_{ij}^{(l,D)}(k+1) &= w_{ij}^{(l,D)}(k) + \Delta w_{ij}^{(l,D)}(k) \\ w_{ij}^{(l,l-1)}(k+1) &= w_{ij}^{(l,l-1)}(k) + \Delta w_{ij}^{(l,l-1)}(k), \quad \forall l, i, j, k = 0, 1, 2, \dots \\ \Delta w_{ij}^{(l,\Lambda)}(k) &= -\alpha \sum_{\nu=1}^L \frac{\partial R_{\text{emp}}^{(\nu)}(\xi)}{\partial w_{ij}^{(l,\Lambda)}} \Big|_{\xi=\xi(k)} \\ \frac{\partial R_{\text{emp}}^{(\nu)}(\xi)}{\partial w_{ij}^{(l,\Lambda)}} &= -2 \sum_{q=1}^{N_L} [\hat{x}_q^{(L)}(\nu) - x_q^{(L)}(\nu)]^2 \frac{\partial x_q^{(L)}(\nu)}{\partial w_{ij}^{(l,\Lambda)}}, \quad \Lambda = l, l-1 \end{aligned} \right\} \quad (2-123)$$

式中  $\alpha > 0$  为步幅。此外，为避免混淆，将式 (2-120) 用于上式时，将其中  $x_i^{(L)}$  等的下标由  $i$  改成为  $q$ 。从下面的推导中可知，上式最右侧的偏微分计算依赖于  $x_q^{(l-1)}(\nu)$  和  $x_q^{(L)}(\nu-1)$  相对于  $w_{ij}^{(l,\Lambda)}$  的偏微分，这样层层下推，可以将式 (2-123) 的计算归结为各  $x_q^{(l')}( \nu)$  和  $x_q^{(l')}( \nu-1), l' = 1 \sim L, q = 1 \sim N_{l'}$  相对于  $w_{ij}^{(l,\Lambda)}, \Lambda = l, l-1$  的偏微分计算  $x_q^{(l')}( \nu)$  和  $x_q^{(l')}( \nu-1)$  的偏微分计算完全取同一方法，所以只需说明前者)。

根据式 (2-119) 可以求得

$$\frac{\partial x_q^{(l')}( \nu)}{\partial w_{ij}^{(l,\Lambda)}} = \frac{\partial x_q^{(l')}( \nu)}{\partial I_q^{(l')}( \nu)} \cdot \frac{\partial I_q^{(l')}( \nu)}{\partial w_{ij}^{(l,\Lambda)}}, \quad l' = 1 \sim L, \forall l, i, j, \Lambda = l, l-1 \quad (2-124)$$

上式右侧第一个偏微分项可按式计算：

$$\frac{\partial x_q^{(l')}( \nu)}{\partial I_q^{(l')}( \nu)} = \begin{cases} 1, & l' = L \\ x_q^{(l')}(1 - x_q^{(l')}( \nu)), & l' = 1 \sim L-1 \end{cases} \quad (2-125)$$

上式右侧第二个偏微分项可以分成下列三种情况来计算。

(1)  $l > l'$

$$\frac{\partial I_q^{(l')}( \nu)}{\partial w_{ij}^{(l,\Lambda)}} = 0, \quad \forall l, i, j, \Lambda = l, l-1$$

(2)  $l = l'$

$$\begin{aligned} \frac{\partial I_q^{(l')}( \nu)}{\partial w_{ij}^{(l,D)}} &= (1 - \delta_{iL}) \left\{ \gamma(\nu) \sum_{p=1}^{N_{l'}} w_{qp}^{(l',l')} \frac{\partial x_p^{(l')}( \nu-1)}{\partial w_{ij}^{(l,D)}} + \delta_q x_j^{(l')}( \nu-1) \right\} \\ \frac{\partial I_q^{(l')}( \nu)}{\partial w_{ij}^{(l,l-1)}} &= (1 - \delta_{iL}) \left\{ \gamma(\nu) \sum_{p=1}^{N_{l'}} w_{qp}^{(l',l')} \frac{\partial x_p^{(l')}( \nu-1)}{\partial w_{ij}^{(l,l-1)}} \right\} + \delta_q x_j^{(l'-1)}( \nu) \end{aligned} \quad (2-126)$$

式中  $\delta_{iL} = 1$  当  $l' = L, \delta_{iL} = 0$  当  $l' \neq L$ 。这意味着当  $l' = L$  时，上列二式的花括弧项被注销。式中  $\delta_q = 1$  当  $q = i$  否则  $\delta_q = 0$ 。两式右侧花括弧中的和式表示过去  $\nu-1$  时刻的偏导数对当前  $(\nu)$  时刻偏导数的影响，所以应称为动态偏导数项。和式前的  $\gamma(\nu)$  称为折扣系数，这是一个略小于或等于 1 的数，它使过去动态导数的影响受到指数衰减或不受衰减。 $\gamma(\nu)$  可随  $\nu$  而略有变化。

(3)  $l < l'$

$$\begin{aligned}\frac{\partial I_q^{(l')}(\nu)}{\partial w_{ij}^{(l,l)}} &= (1 - \delta_{lL}) \left\{ \gamma(\nu) \sum_{p=1}^{N_{l'}} w_{qp}^{(l',l')} \frac{\partial x_p^{(l')}(\nu-1)}{\partial w_{ij}^{(l,l)}} \right\} + \sum_{r=1}^{N_{l'-1}} w_{qr}^{(l',l'-1)} \frac{\partial x_r^{(l'-1)}(\nu)}{\partial w_{ij}^{(l,l)}} \\ \frac{\partial I_q^{(l')}(\nu)}{\partial w_{ij}^{(l,l-1)}} &= (1 - \delta_{lL}) \left\{ \gamma(\nu) \sum_{p=1}^{N_{l'}} w_{qp}^{(l',l')} \frac{\partial x_p^{(l')}(\nu-1)}{\partial w_{ij}^{(l,l-1)}} \right\} + \sum_{r=1}^{N_{l'-1}} w_{qr}^{(l',l'-1)} \frac{\partial x_r^{(l'-1)}(\nu)}{\partial w_{ij}^{(l,l-1)}}\end{aligned}\quad (2-127)$$

注意，如果网络的输出层 ( $l = L$ ) 也存在延时一拍的层内全反馈时，只需将式 (2-126) 和式 (2-127) 等号右侧的  $(1 - \delta_{lL})$  换成 1，即可用它们进行计算。这样，只要给定下列初值：

$$x_q^{(l')}(0) = 0, \quad \frac{\partial x_q^{(l')}(0)}{\partial w_{ij}^{(l,l)}} = 0, \quad \frac{\partial x_q^{(l')}(0)}{\partial w_{ij}^{(l,l-1)}} = 0, \quad \forall l', l, q, i, j$$

就可以按时序  $\nu = 1, 2, \dots$  计算出所有需要的偏导数。在此算法中，任一时刻的偏导数计算都依赖于前一时刻的偏导数计算，所以称为动态梯度算法。

### 2.13.4 全反馈 RNN 的动态梯度学习算法<sup>[143]</sup>

当 RNN 不仅包括各层的内反馈环路，还包括输出层的各个信号延时一个节拍后全部反馈回来，与外部输入信号一起构成网络的输入，这种网络称为全反馈 RNN。在此网络中任意一层的神经元输出都受网络中任意一个权的影响，因此相应的偏导数计算应修正为下列形式（这里给出的是输出层具有层内反馈情形下的计算公式）：

$$\begin{aligned}\frac{\partial I_q^{(l')}(\nu)}{\partial w_{ij}^{(l,l)}} &= \gamma(n) \sum_{p=1}^{N_{l'}} w_{qp}^{(l',l')} \frac{\partial x_p^{(l')}(\nu-1)}{\partial w_{ij}^{(l,l)}} + \sum_{r=1}^{N_{l'-1}} w_{qr}^{(l',l'-1)} \frac{\partial x_r^{(l'-1)}(\nu)}{\partial w_{ij}^{(l,l)}} + \delta_{qi} \delta_{l'l} x_j^{(l')}(\nu-1), \\ \frac{\partial I_q^{(l')}(\nu)}{\partial w_{ij}^{(l,l-1)}} &= \gamma(n) \sum_{p=1}^{N_{l'}} w_{qp}^{(l',l')} \frac{\partial x_p^{(l')}(\nu-1)}{\partial w_{ij}^{(l,l-1)}} + \sum_{r=1}^{N_{l'-1}} w_{qr}^{(l',l'-1)} \frac{\partial x_r^{(l'-1)}(\nu)}{\partial w_{ij}^{(l,l-1)}} + \\ &\quad \delta_{qi} \delta_{l'l} x_j^{(l'-1)}(\nu), \quad \forall q, l, l', i, j\end{aligned}\quad (2-128)$$

计算的初始条件与上一小节相同，不再重复。应指出 对于  $l' = 1$ ，在计算上式中所需的有关  $l' - 1 = 0$  的各项应按下式计算：

$$\begin{aligned}x_q^{(0)}(\nu) &= x_q(\nu), \quad q = 1 \sim N \\ \frac{\partial x_q^{(0)}(\nu)}{\partial w_{ij}^{(l,l)}} &= 0, \quad \forall i, j, l, \Delta = l, l-1, \quad q = 1 \sim N\end{aligned}$$

以上诸项相应于外部对网络的输入。

$$\begin{aligned}x_q^{(0)}(\nu) &= x_{q-N}^{(L)}(\nu-1), \quad q = N+1 \sim N+N_L \\ \frac{\partial x_q^{(0)}(\nu)}{\partial w_{ij}^{(l,l)}} &= \frac{\partial x_{q-N}^{(L)}(\nu-1)}{\partial w_{ij}^{(l,l)}}, \quad \forall i, j, l, \Delta = l, l-1, \quad q = N+1 \sim N+N_L\end{aligned}$$

以上诸项相应于网络的输出层各信号延迟一个节拍后反馈回来对网络的输入。

### 2.13.5 EKF 算法<sup>[143,158,159,160,161,166]</sup>

EKF 算法将一个 RNN 的训练问题作为一个非线性动力系统的动态参数估计问题来处理，即随时序  $\nu$  不断对  $\xi(\nu)$  进行重估，使得 RNN 的输出列向量  $\mathbf{X}^{(L)}(\nu)$  与理想输出列向量  $\mathbf{X}^{(L)}(\nu)$  之间的某种误差随  $\nu$  的增加而逐渐减小。EKF 用于函数逼近及模式分类等问题

时，其性能优于基于最陡下降的 BP 算法且收敛速度更快，便于实现在线自适应。下面依次讨论作为基础的 GEKF 和效率更高的 DEKF 两种算法。

### 1. GEKF 算法（全局 EKF 算法）

G 是 global 的缩写。设  $\boldsymbol{\eta}(\nu) = \hat{\mathbf{X}}^{(L)}(\nu) - \mathbf{X}^{(L)}(\nu)$  这是时刻  $\nu$  的  $N_L$  维误差列向量。现在定义一个目标误差函数  $E(\nu)$  如下：

$$E(\nu) = \frac{1}{2} \boldsymbol{\eta}^T(\nu) \mathbf{S}(\nu) \boldsymbol{\eta}(\nu) \quad (2-129)$$

其中  $\mathbf{S}(\nu)$  是一个  $N_L \times N_L$  维方阵，称为加权阵，其元素皆非负。如果  $\mathbf{S}(\nu)$  是一个对角元素为 1 而其他元素为 0 的单位阵，则  $E(\nu)$  退化为式 (2-120) 定义的瞬时风险函数。 $\mathbf{S}(\nu)$  的其他可能定义将在下文中举例说明。

设网络中权和阈值的总数为  $Q$ ，它们构成一个表征系统状态的  $Q$  维列向量  $\boldsymbol{\xi}(\nu)$ （可简称为状态）。这是一个随  $\nu$  改变而不断得到重估的随机向量，其各分量之间的协方差值构成一个  $Q \times Q$  维协方差阵  $\mathbf{P}(\nu)$ 。若  $\boldsymbol{\xi}(\nu)$  的最佳值为  $\boldsymbol{\xi}(\nu)$ （使  $E(\nu)$  最小）那么重估算法应使前者随  $\nu$  的增加而逼近后者。设  $\mathbf{X}^{(L)}(\nu)$  的  $N_L$  个分量相对于  $\boldsymbol{\xi}(\nu)$  的  $Q$  个分量的偏导数构成一个  $Q \times N_L$  维矩阵  $\mathbf{H}(\nu)$ （ $\mathbf{H}(\nu)$  各分量可以用 2.13.3 节所述方法计算），这样，可以采取下列算法，求得每个时刻  $\nu$  的最佳状态向量估值  $\boldsymbol{\xi}(\nu)$ 。

首先，在时刻  $\nu = 0$  设置最佳权向量初值  $\boldsymbol{\xi}(0)$  及其相关阵初值  $\mathbf{P}(0)$ ，前者的各分量皆取绝对值小于 1 的随机值，后者取为对角阵且每一对角元素是量级为 100 的正数。

其次，对于时刻  $\nu = 0, 1, 2, \dots$  按下列的 GEKF 算法对  $\boldsymbol{\xi}(\nu+1)$  和  $\mathbf{P}(\nu+1)$  进行重估。

(1) 计算一个  $N_L \times N_L$  维方阵  $\mathbf{A}(\nu)$ ，

$$\mathbf{A}(\nu) = [(\alpha(\nu)\mathbf{S}(\nu))^{-1} + \mathbf{H}^T(\nu)\mathbf{P}(\nu)\mathbf{H}(\nu)]^{-1} \quad (2-130)$$

其中  $\alpha(\nu)$  为取非负值的步幅函数， $\mathbf{S}(\nu)$  由算法使用者自定（详见下文）， $\mathbf{H}(\nu)$  的各个分量由上节所述的动态求导算法求得。这一步计算涉及一个  $N_L \times N_L$  维方阵求逆的问题。

(2) 计算一个  $Q \times N_L$  维矩阵  $\mathbf{K}(\nu)$ ，

$$\mathbf{K}(\nu) = \mathbf{P}(\nu)\mathbf{H}(\nu)\mathbf{A}(\nu) \quad (2-131)$$

此矩阵称为 Kalman 增益阵。

(3) 计算重估的参数向量  $\boldsymbol{\xi}(\nu+1)$  和重估的相关阵  $\mathbf{P}(\nu+1)$ ，

$$\boldsymbol{\xi}(\nu+1) = \boldsymbol{\xi}(\nu) + \mathbf{K}(\nu)\boldsymbol{\eta}(\nu) \quad (2-132)$$

$$\mathbf{P}(\nu+1) = \mathbf{P}(\nu) - \mathbf{K}(\nu)\mathbf{H}^T(\nu)\mathbf{P}(\nu) + \boldsymbol{\Pi}(\nu) \quad (2-133)$$

$\boldsymbol{\Pi}(\nu)$  是一个  $Q \times Q$  维对角协方差阵，其作用在于在 Kalman 递归滤波过程中引入人工噪声，以避免计算过程中的算法数值发散以及避免陷入劣质局部极小点，详见文献 [162]。 $\mathbf{S}(\nu)$  的一种可能的设置方案 [163] 是，当某个时刻  $\nu$  的  $x_i^{(L)}(\nu)$  与  $\hat{x}_i^{(L)}(\nu)$  之间的误差低于某一门限时，与该项有关的各误差项及导数项不应参与迭代计算。这时可以令  $\mathbf{S}(\nu)$  中与该项有关的各分量置为 0。当  $\mathbf{S}(\nu)$  为非单位阵时，为求  $\mathbf{A}(\nu)$ ，需进行两次矩阵求逆计算（式 (2-129)）。如进行适当变换，可以减少一次求逆计算。令

$$\boldsymbol{\eta}^*(\nu) = \mathbf{S}^{\frac{1}{2}}(\nu)\boldsymbol{\eta}(\nu) \quad (2-134)$$



$$E(\nu) = \frac{1}{2}(\boldsymbol{\eta}^*(\nu))^T \boldsymbol{\eta}^*(\nu) \quad (2-135)$$

$$\mathbf{H}^*(\nu) = \mathbf{H}(\nu) \mathbf{S}^{\frac{1}{2}}(\nu) \quad (2-136)$$

$\mathbf{H}^*(\nu)$  可以通过  $\boldsymbol{\eta}^*(\nu)$  各分量对  $\boldsymbol{\xi}(\nu)$  各分量求偏导获得, 这样 GEKF 可以按下列诸式进行计算。

$$(1) \quad \mathbf{A}^*(\nu) = [\alpha^{-1}(\nu) \mathbf{I}(\nu) + (\mathbf{H}^*(\nu))^T \mathbf{P}(\nu) \mathbf{H}^*(\nu)]^{-1} \quad (2-137)$$

其中  $\mathbf{I}$  是一个  $N_L \times N_L$  维单位阵;

$$(2) \quad \mathbf{K}^*(\nu) = \mathbf{P}(\nu) \mathbf{H}^*(\nu) \mathbf{A}^*(\nu) \quad (2-138)$$

$$(3) \quad \boldsymbol{\xi}(\nu+1) = \boldsymbol{\xi}(\nu) + \mathbf{K}^*(\nu) \boldsymbol{\eta}^*(\nu) \quad (2-139)$$

$$\mathbf{P}(\nu+1) = \mathbf{P}(\nu) - \mathbf{K}^*(\nu) (\mathbf{H}^*(\nu))^T \mathbf{P}(\nu) + \boldsymbol{\Pi}(\nu) \quad (2-140)$$

按上列诸式计算时, 每个时刻  $\nu$  的 GEKF 计算量级为  $O(N_L Q^2)$  存储量级为  $O(Q^2)$ 。

## 2. DEKF(解耦 EKF 算法)

D 是 decoupled 的缩写。一般的规模稍大的 RNN 中常包含数百(甚至上千)参数, 即  $Q$  的量级为几百甚至上千, 这时 GEKF 的计算及存储量将非常大。DEKF 将全部参数分成若干组, 每个组内的权存在相关性而各组之间不相关, 这样就可以得到一种计算及存储量大大减少的算法。这称为解耦。

实际做法是将每个神经元的各个输入权参数(包括阈值)归为一个组。每个神经元的各权参数用列向量  $\boldsymbol{\xi}_i(\nu)$  表示, 其维数为  $Q_i, i=1 \sim G, G$  是网络中全部神经元的数量。在假定各个  $\boldsymbol{\xi}_i(\nu)$  互不相关的条件下, 可以将一个总向量  $\boldsymbol{\xi}(\nu)$  的重估计算拆散为  $G$  个分向量  $\boldsymbol{\xi}_i(\nu)$ , 独立分别进行重估的计算。为此先作下列定义。已知  $\boldsymbol{\eta}^*(\nu) = \mathbf{S}^{\frac{1}{2}}(\nu) \boldsymbol{\eta}(\nu)$ ,  $\boldsymbol{\eta}^*(\nu)$  各分量相对于  $\boldsymbol{\xi}(\nu)$  各分量的偏导数构成一个  $Q_i \times N_L$  维矩阵  $\mathbf{H}_i^*(\nu)$ 。  $Q_i \times Q_i$  方阵  $\mathbf{P}_i(\nu)$  是  $\boldsymbol{\xi}_i(\nu)$  各分量的自关阵。  $Q_i \times N_L$  维矩阵  $\mathbf{K}_i^*(\nu)$  是  $\mathbf{H}_i^*(\nu)$  的卡尔曼滤波增益阵。这样, 可以构成下列重估算法。

$$(1) \quad \mathbf{A}^*(\nu) = \left[ \alpha^{-1}(\nu) \mathbf{I} + \sum_{i=1}^G (\mathbf{H}_i^*(\nu))^T \mathbf{P}_i(\nu) \mathbf{H}_i^*(\nu) \right]^{-1} \quad (2-141)$$

这里  $\mathbf{A}^*(\nu)$  仍是一个  $N_L \times N_L$  维方阵;

$$(2) \quad \mathbf{K}_i^*(\nu) = \mathbf{P}_i(\nu) \mathbf{H}_i^*(\nu) \mathbf{A}^*(\nu), \quad i=1 \sim G \quad (2-142)$$

$$(3) \quad \hat{\boldsymbol{\xi}}_i(\nu+1) = \hat{\boldsymbol{\xi}}_i(\nu) + \mathbf{K}_i^*(\nu) \boldsymbol{\eta}^*(\nu), \quad i=1 \sim G \quad (2-143)$$

$$\mathbf{P}_i(\nu+1) = \mathbf{P}_i(\nu) - \mathbf{K}_i^*(\nu) (\mathbf{H}_i^*(\nu))^T \mathbf{P}_i(\nu) + \boldsymbol{\Pi}_i(\nu), \quad i=1 \sim G \quad (2-144)$$

$\boldsymbol{\Pi}_i(\nu)$  是  $Q_i \times Q_i$  维对角协方差阵(人工噪声阵)。

DEKF 的计算量量级为  $O\left(N_L^2 \sum_{i=1}^G Q_i + N_L \sum_{i=1}^G Q_i^2\right)$ , 存储量量级为  $O\left(\sum_{i=1}^G Q_i^2\right)$ 。一般情况下  $N_L$  是一个很小的数,  $Q_i$  也是不大的数, 所以这种算法的计算量和存储量均较 GEKF 算法要低得多。此外应指出, 虽然以上的讨论是针对 RNN 的, 但也能适用于一般的 MLFN。

## 2.13.6 RNN 的应用

RNN 应用潜力最大的领域之一是非线性动力系统的辨识与控制。在下一节将对

MLFN 在这一领域的应用作深入探讨，这一节只对 RNN 在此领域的应用特点作初步讨论。

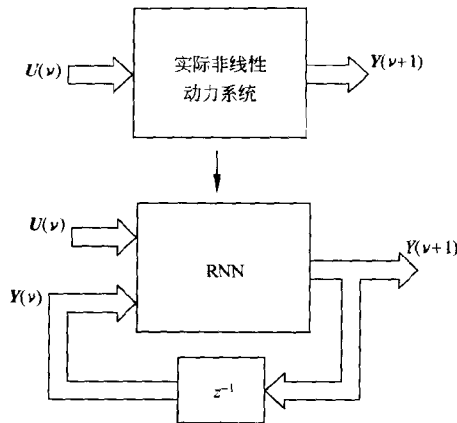


图 2-30 实际非线性动力系统的全反馈 RNN 示意图

设有一个实际的离散时域非线性动力系统，其输入为  $N$  维向量  $U(\nu)$  其输出为  $M$  维向量  $Y(\nu)$ 。系统在  $\nu + 1$  时刻的输出取决于  $\nu$  时刻的输入和输出，这可以表示为

$$Y(\nu + 1) = F(U(\nu), Y(\nu)) \tag{2-145}$$

$F(\cdot)$  是一个未知的非线性函数。为了对此实际系统建模，可以采用图 2-30 所示的全反馈 RNN。通过采自实际系统的训练集  $\{U(\nu), Y(\nu + 1)\}$  对 RNN 进行训练后将使 RNN 的输出  $Y(\nu + 1)$  逼近于  $Y(\nu + 1)$ 。这样就能用该 RNN 代替实际系统进行各种离线（脱离现场）研究。此 RNN 即构成一个原系统的辨识模型，这里解决的是一个非线性系统辨识问题，该 RNN 称为一个 ID 网络（ID 是 identification 的前两个字母）。

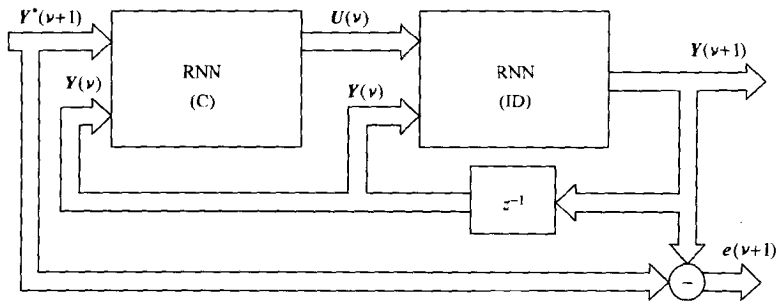


图 2-31 图 2-30 例子的实施方案

系统控制的目的是设计一个控制器（C 网络）它产生一个控制信号  $U(\nu)$  输往实际被控系统 使得其输出  $Y(\nu + 1)$  逼近于一个预定的函数  $Y^*(\nu + 1)$ 。为此可以采用图 2-31 所示的实施方案，其中的 ID 网络用一个 RNN 来对实际被控系统按上述方法建模。C 网络是另一个 RNN 其输入为  $Y(\nu)$  和  $\nu + 1$  时刻预定达到的被控系统输出  $Y^*(\nu + 1)$  其输出为  $U(\nu)$ 。当 ID 网络训练完毕后，将 C 网络和 ID 网络合在一起训练，这时二者组合成一个大 RNN。在训练中要求  $Y(\nu + 1)$  与  $Y^*(\nu + 1)$  之差  $e(\nu)$  的均方值达到最小。为此可以采用动

态梯度算法或 DEKF 算法。所涉及的  $Y(\nu+1)$  相对于两个 RNN 中各个权的偏导数计算仍可用 2.13.2 小节和 2.13.3 小节中所述的方法。应指出，在对 C 网络进行训练时，虽然要用到 ID 网络中各个权的偏导数，但是这些权并不变值，只有 C 网络中的权在训练中变化，以使得  $e(\nu)$  的均方值最小。

文献[143]、[161]报道了将 RNN 用于汽车控制时的各种实例，着重讨论了汽车发动机的空转速度控制这样一个高度非线性的控制问题，给出了 RNN 的设计实例和各种训练算法的效果（包括动态梯度算法和 DEKF 算法）。文献[140]给出了将 RNN 用于能源工业的一个实例。能源工业中非常重要的一个环节是高压循环水（初级环）和低压循环水（次级环）之间的热交换，运行中存在着液态和气态的二相混合。如果此系统控制不当，将引起意外事故甚至停机，从而造成严重损失。RNN 可以用于该系统的监督、故障早期诊断和自适应过程控制。实验结果表明，采用 RNN 时的效果明显优于采用静态映射的 MLFN。因此，虽然 RNN 的训练中计算各个偏导数的计算量很大，这个代价是值得的。

RNN 的另一个重要应用领域是时间序列预测。以一维情况为例，设有一个一维变量的离散时间序列  $y(\nu), \nu = \dots, -1, 0, 1, 2, \dots$ 。对于任何一个时刻  $\nu$ ，若  $y(\nu+1)$  是未知的，而  $\nu$  和  $\nu$  以前的  $y(\nu)$  值是已知的，记为  $\hat{y}(\nu), \hat{y}(\nu-1), \hat{y}(\nu-2), \dots$ ，则可以用这些已知值对未知的  $y(\nu+1)$  进行预测。如果还有一个  $y(\nu)$  与之有依从关系的已知向量序列  $X(\nu)$ ，则可用  $X(\nu), X(\nu-1), X(\nu-2), \dots$  参与  $y(\nu+1)$  的预测。如果采用线性分析的方法就导致线性 MA 模型，即 LPC 分析模型<sup>[32]</sup>。采用 TDNN(NMA 模型)可以改善预测效果。采用 RNN(NARMA 模型)则可进一步改善。时间序列预测的应用方面很多，例如能源消耗预测<sup>[141]</sup>和包括股价指数预测、汇率预测、期货指数预测在内的资本市场和金融领域中的各项预测问题。后者由于市场和金融的全球化日益加剧而显得尤其重要。股指和期指预测常用的神经网络是 TONN、RNN、PNN(概率神经网络)和 FNN(模糊神经网络)。文献[164]比较了前三者在股指预测中的性能，其中 RNN 的效果最好。FNN 在资本和金融市场预测中的重要性日益增加，在后文 5.7.1 节中将回到这个题目作更详细的讨论。

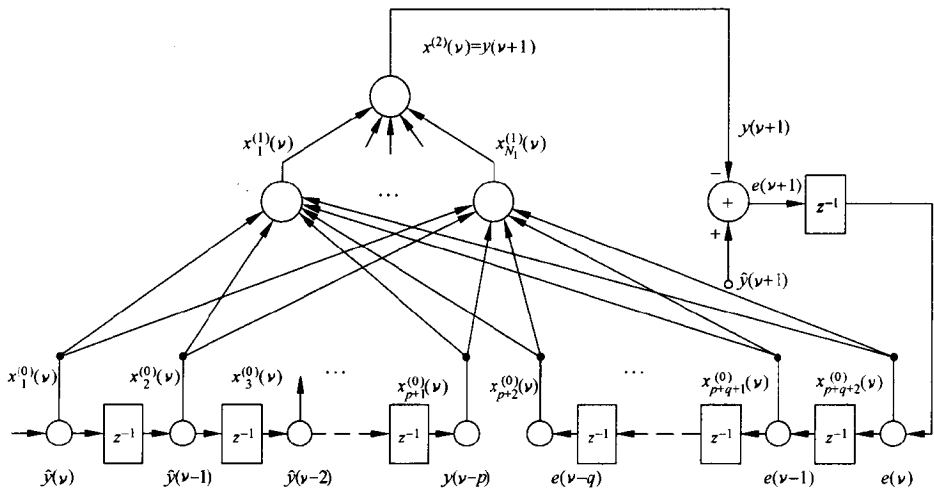


图 2-32 一种 RNN 网络结构

为了提高时间序列预测的精度，还可以采取图 2-32 所示的一种结构。这时除了以  $\hat{y}(\nu), \hat{y}(\nu-1), \dots, \hat{y}(\nu-P)$  作为网络输入来预测  $y(\nu+1)$  外，还将预测误差  $\hat{\epsilon}(\nu), \hat{\epsilon}(\nu-1), \dots, \hat{\epsilon}(\nu-Q)$  作为网络输入。其中  $\hat{\epsilon}(\nu) = \hat{y}(\nu) - y(\nu)$ ， $P$  和  $Q$  是两个正常数。这实际上是一种 RNN<sup>[141]</sup>。

RNN 的用途很广，除前述例子外，还包括在自适应信道均衡中的应用<sup>[142]</sup>、音素概率估计<sup>[144]</sup>等。有关理论可进一步参考文献 [143]、[145]、[146] 等。有关 RNN 的稳定性问题是一个仍在研究中的课题。

## 2.14 MLFN 应用举例之一 —— 在非线性能力系统中的应用

在工业控制、CIMS、化工、能源、机器人乃至社会和经济的各个领域中都存在着各式各样的非线性动力系统。将这些系统用一个线性系统来近似表示，可以使分析和计算大为简化，但是采用线性模型时只有当运行范围有限时才能较为符合实际情况，否则理论分析的结果与实际系统的运行情况差异颇大。采用非线性系统模型更符合实际情况，但是计算与分析都存在很大困难。人工神经网络提供了一个建立非线性动力系统模型的极有吸引力的新途径。它的主要用途包括：建立实际系统的非线性模型，以便于对实际系统进行离线的模拟、分析和设计等研究；实现对实际的非线性动力系统的调节、控制和跟踪，从而实现全部生产过程的自动控制、自适应控制和计算机控制；实现实际系统中的参数提取和故障诊断；实现对系统状态的估计；等等。已进行的大量研究证明，采用人工神经网络（几乎全是 MLFN 和 RNN）可以有效地改善上述各个方面的目标。

### 2.14.1 离散时域非线性动力系统简介

#### 1. 系统运行方程

为简化计，只讨论单输入单输出（SISO）系统。对于离散时间  $\nu$  系统可以用 3 个变量描述：输入  $u(\nu)$ 、输出  $y(\nu)$ 、系统状态  $\mathbf{X}(\nu)$ ；前二者为实变量，第三者为一个  $N$  维行向量（其各分量为实变量），那么，一个系统的运行可用下列方程描述并记之为  $\zeta$ ：

$$\zeta: \begin{cases} \mathbf{X}(\nu+1) = \mathbf{f}(\mathbf{X}(\nu), u(\nu)) \\ y(\nu) = h(\mathbf{X}(\nu)) \end{cases} \quad (2-146)$$

其中  $\mathbf{f}(\cdot)$  实现  $\mathbb{R}^{N+1} \rightarrow \mathbb{R}^N$  映射， $h(\cdot)$  则实现  $\mathbb{R}^N \rightarrow \mathbb{R}^1$  映射。

此系统的一种最简单的特殊情况是线性非时变系统  $\zeta_L$ ，这时式 (2-146) 退化为下列形式：

$$\zeta_L: \begin{cases} \mathbf{X}^T(\nu+1) = \mathbf{A} \mathbf{X}^T(\nu) + \mathbf{b}^T u(\nu) \\ y(\nu) = \mathbf{C} \mathbf{X}^T(\nu) \end{cases} \quad (2-147)$$

其中  $\mathbf{A}$  是  $N \times N$  维矩阵， $\mathbf{b}$  和  $\mathbf{C}$  都是  $N$  维行向量。它们的各个分量皆取实数值。

#### 2. 非线性动力系统研究中的主要问题

就一个实际的系统而言，其输入  $u(\nu)$  和输出  $y(\nu)$  较易于通过测量而取得，而状态  $\mathbf{X}(\nu)$  涉及系统内部的情况，在系统正在运行的过程中，直接对其进行监测和控制较为困难。这样，无论是对于一个具体系统进行操作、监控以保障其正常运行，还是通过模拟用仿

真系统进行研究，都需要对系统进行若干项研究。下面列举其中的一部分。

(1) 状态估计。对于任何时刻  $\nu$  若  $y(\nu), y(\nu-1), \dots$  以及  $u(\nu-1), u(\nu-2), \dots$  为已知, 对  $X(\nu)$  作出估计称为状态估计。一个状态估计器可以称为观察器, 它既可用于系统仿真, 又可用于具体系统的实时控制、过程参数提取或故障诊断。

(2) 系统辨识。对于任何  $\nu$  若  $y(\nu), y(\nu-1), \dots$  以及  $u(\nu), u(\nu-1), \dots$  为已知, 对系统在  $\nu+1$  时刻的输出  $y(\nu+1)$  进行估计, 称为系统辨识。它可用于系统仿真或实际系统输出的在线实时预测和控制等。

(3) 系统控制。可分成两方面。第一方面, 使系统状态保持在一个所需的平衡点附近; 即使有干扰使其暂时脱离此平衡点, 也能很快回到此点并稳定下来。这称为系统的调节功能( regulation)。第二方面, 对于一个具体的系统, 设计一个反馈控制器, 使得系统的输出沿着一条预置的轨迹运行。这称为系统的跟踪功能( tracking ) 或称为控制功能。

### 3. 非线性动力系统的可观性、可控性和稳定性

(1) 可观性。设系统于离散时间起点  $\nu = 0$  所处状态为  $X(0)$  在  $\nu = 0, 1, \dots, l-2$  以及  $l-1$  诸时刻系统的输入和输出分别用  $l-1$  维行向量  $U_{l-1} = [u(0), \dots, u(l-2)]$  和  $l$  维行向量  $Y_l = [y(0), \dots, y(l-1)]$  表示。 $Y_l$  是  $X(0)$  和  $U_{l-1}$  的函数 可表示为  $Y_l[X(0), U_{l-1}]$  若  $X(0)$  取两个不同的状态  $X_1$  和  $X_2$  时 有  $Y_l[X_1, U_{l-1}] \neq Y_l[X_2, U_{l-1}]$  则称系统是可观的。其中  $l$  是某个正整数。

(2) 可控性。对于系统中的任何两个状态,  $X_1$  和  $X_2$ , 如果总存一个有限长度输入序列, 能够使系统从状态为  $X_1$  转移到状态为  $X_2$  则称此系统为可控的。

(3) 稳定性。若存在状态  $\bar{X}$  和输入  $\bar{u}$  当  $X(\nu) = \bar{X}$  且  $u(\nu) = \bar{u}$  时  $X(\nu+1) = \bar{X}$  这可以简洁地表示为  $\bar{X} = f(\bar{X}, \bar{u})$  则  $\bar{X}$  称为系统的一个平衡态。 $\bar{X} = 0$  (原点), 当  $\bar{u} = 0$  时, 总是系统的一个平衡态。总可以对此平衡态进行讨论而不失一般性。

若对于平衡态  $\bar{X}$  的任何一个邻域  $\Omega$  存在  $\bar{X}$  的某个邻域  $\theta, \theta \subset \Omega$  当  $X(0) \in \theta$  时,  $X(\nu) \in \Omega, \forall \nu$ 。则称  $\bar{X}$  是稳定平衡态。

对于平衡态  $\bar{X} = 0$ , 若可选择邻域  $\theta$  当  $X(0) \in \theta$  时,  $\lim_{\nu \rightarrow \infty} X(\nu) = 0$  则称原点这个平衡态是渐近稳定的。若能在有限步内使系统状态从  $X(0)$  运行到  $0$  则称为有限步稳定的。如果  $\theta$  与  $R^N$  重合, 则称为全局稳定的; 反之, 称为局部渐近稳定的。若对于一个系统能通过选择反馈函数  $u(\nu) = g(X(\nu))$  使得  $X = 0$  成为渐近稳定的, 则系统称为可稳定化的。

## 2. 14. 2 MLFN 在非线性动力系统状态估计中的应用

作为一个出发点, 首先讨论一个线性非时变系统。按照式 ( 2-147) 若系统在时刻  $\nu$  处于状态  $X(\nu)$  则  $\nu + N - 1$  时刻系统所处状态为

$$X^T(\nu + N - 1) = A^{N-1} X^T(\nu) + A^{N-2} b^T u(\nu) + \dots + b^T u(\nu + N - 2) \quad (2-148)$$

系统在时刻  $\nu, \nu + 1, \dots, \nu + N - 1$  的输出为

$$y(\nu) = C X^T(\nu)$$

$$y(\nu + 1) = C A X^T(\nu) + C b^T u(\nu)$$

$$\dots\dots \dots\dots$$

$$\dots\dots \dots\dots$$

$$y(\nu + N - 1) = C A^{N-1} X^T(\nu) + C A^{N-2} b^T u(\nu) + \dots + C b^T u(\nu + N - 2)$$

上列诸式可以写成下列的矩阵形式方程式

$$\underbrace{\begin{bmatrix} y(\nu) \\ y(\nu+1) \\ \vdots \\ y(\nu+N-1) \end{bmatrix}}_{\mathbf{Y}_N^T(\nu)} = \underbrace{\begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{N-1} \end{bmatrix}}_{\mathbf{M}_0} \mathbf{X}^T(\nu) + \underbrace{\begin{bmatrix} 0 & 0 & \cdots & 0 \\ \mathbf{C}\mathbf{b}^T & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{CA}^{N-2}\mathbf{b}^T & \mathbf{CA}^{N-3}\mathbf{b}^T & \cdots & \mathbf{C}\mathbf{b}^T \end{bmatrix}}_{\mathbf{M}_1} \underbrace{\begin{bmatrix} u(\nu) \\ u(\nu+1) \\ \vdots \\ u(\nu+N-2) \end{bmatrix}}_{\mathbf{U}_{N-1}^T(\nu)} \quad (2-149)$$

如果此系统是可观的,则可以由  $\mathbf{Y}_N(\nu)$  和  $\mathbf{U}_{N-1}(\nu)$  惟一地确定  $\mathbf{X}(\nu)$  其充要条件为  $\mathbf{M}_0$  非奇。若  $\mathbf{M}_0$  非奇 则称之为可观阵。若  $\mathbf{M}_0, \mathbf{M}_1$  已知且  $\mathbf{M}_0$  非奇 则根据  $\mathbf{Y}_N(\nu)$  和  $\mathbf{U}_{N-1}(\nu)$  由上式可求解得  $\mathbf{X}(\nu)$ 。再将  $\mathbf{X}(\nu)$  代入式 (2-148) 即可求得  $\mathbf{X}(\nu+N-1)$  它表示为  $y(\nu), y(\nu+1), \dots, y(\nu+N-1)$  以及  $u(\nu), u(\nu+1), \dots, u(\nu+N-2)$  的线性组合。再进行适当的时间变量变换,即可将系统在任一时刻  $\nu$  的状态  $\mathbf{X}(\nu)$  表示为  $y(\nu), y(\nu-1), \dots, y(\nu-N+1)$  以及  $u(\nu-1), u(\nu-2), \dots, u(\nu-N+1)$  的线性组合。这可以表示为

$$\mathbf{X}(\nu) = \varphi[\mathbf{Y}_N(\nu-N+1), \mathbf{U}_{N-1}(\nu-N+1)] \quad (2-150)$$

其中  $\varphi[\cdot]$  是一个线性函数。如果式 (2-149) 中的  $\mathbf{A}, \mathbf{C}, \mathbf{b}$  为已知,  $\varphi[\cdot]$  易于求得。如果它们是未知的,这时  $\mathbf{X}(\nu)$  的估计归结为一个线性估计问题(注意,上文中的  $N$  是  $\mathbf{X}(\nu)$  的维数  $\mathbf{Y}_N(\nu-N+1) = [y(\nu-N+1), \dots, y(\nu)], \mathbf{U}_{N-1}(\nu-N+1) = [u(\nu-N+1), \dots, u(\nu-1)]$ 。

对于非线性系统,可观性的问题要复杂得多。如果依据一个长度为  $l$  的输出序列  $\mathbf{Y}_l(\nu-l+1)$  和一个长度为  $l-1$  的输入序列  $\mathbf{U}_{l-1}(\nu-l+1)$  可以惟一地确定系统的状态  $\mathbf{X}(\nu)$  则称该系统是强可观的。对于一个线性系统 如果设  $l = N$  且  $\mathbf{M}_0$  为可观阵,则该系统是强可观的且  $l = N$ 。若系统是非线性的,在原点为平衡点的情况,则该系统  $\zeta$ (见式 (2-146)) 在原点附近可以近似为如下一个线性系统  $\zeta_L$ (见文献[119]):

$$\zeta_L: \begin{cases} \delta \mathbf{X}^T(\nu+1) = \hat{\mathbf{A}} \delta \mathbf{X}^T(\nu) + \hat{\mathbf{b}}^T \delta u(\nu) \\ \delta y(\nu) = \hat{\mathbf{C}} \delta \mathbf{X}^T(\nu) \end{cases} \quad (2-151)$$

其中  $\delta \mathbf{X}, \delta u, \delta y$  表示  $\mathbf{X} = \mathbf{0}, u = 0, y = 0$  附近的变量。其中  $\hat{\mathbf{A}}$  为函数  $f(\cdot)$  在原点附近相对于  $\mathbf{X}$  的  $N \times N$  维雅可比矩阵,这可以表示为

$$\hat{\mathbf{A}} = \mathbf{f}_x \bigg|_{\mathbf{X}=\mathbf{0}, u=0} = (\partial f_i / \partial x_j) \bigg|_{\mathbf{X}=\mathbf{0}, u=0}$$

类似的,  $\hat{\mathbf{b}}$  为  $f(\cdot)$  在原点附近相对于  $u$  的导数向量(一个  $N$  维行向量),  $\hat{\mathbf{C}}$  为  $h(\cdot)$  在原点附近相对于  $\mathbf{X}$  的梯度  $N$  维行向量,它们可以表示为

$$\hat{\mathbf{b}} = \mathbf{f}_u \bigg|_{\mathbf{X}=\mathbf{0}, u=0}$$

$$\hat{\mathbf{C}} = \nabla_x h \bigg|_{\mathbf{X}=\mathbf{0}}$$

文献[119]证明 如果  $\zeta_L$  是可观的,则  $\zeta$  在原点附近是局部强可观的。这样,当  $\mathbf{X}(\nu)$  在  $\mathbf{0}$  附近时,可用下列公式对其进行估计。

$$\mathbf{X}(\nu) = \Phi[\mathbf{Y}_N(\nu-N+1), \mathbf{U}_{N-1}(\nu-N+1)] \quad (2-152)$$

其中  $\Phi(\cdot)$  是一个非线性函数,  $\hat{X}$  表示对  $X$  的估计。有了上述的理论准备后, 就可以采用一个神经网络  $NN_\Phi$  来实现  $\Phi(\cdot)$ , 可称之为一个状态估计器或观察器, 其结构及训练框图如图 2-33 所示。图中虚线框出部分表示一个实际系统,  $Y_N(\nu-N+1)$  和  $U_{N-1}(\nu-N+1)$  由输出  $y(\nu)$  和输入  $u(\nu)$  经过两条延时线 TDL 获取。 $NN_\Phi$  表示一个神经网络, 其映射函数为  $NN_\Phi[\cdot]$ 。 $X(\nu)$  的估值  $\hat{X}(\nu)$  由  $NN_\Phi$  取得:

$$\hat{X}(\nu) = NN_\Phi[Y_N(\nu-N+1) \quad U_{N-1}(\nu-N+1)]$$

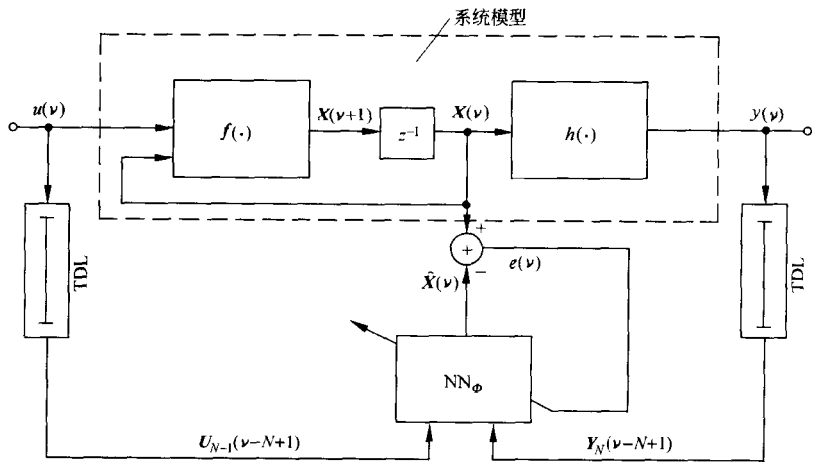


图 2-33  $NN_\Phi$  网络的结构和训练框图

如果能测得实际系统的状态  $X(\nu)$  则可求得误差信号  $e(\nu)$ ,  $e(\nu) = X(\nu) - \hat{X}(\nu)$ 。 $NN_\Phi$  可取本章讨论过的各种形式 MLFN 其中最常用的是 Sigmoid 函数（隐层）和线性函数（输出层）神经元构成的 MLFN。训练数据  $Y_N(\nu-N+1), U_{N-1}(\nu-N+1), X(\nu), \nu = 1, 2, \dots$ , 必须对实际系统进行现场实时采集。而对  $NN_\Phi$  的训练可以离线非实时进行。在训练完毕后, 将在线实时采集到的  $y(\nu)$  和  $u(\nu)$  送入 TDL 再将 TDL 的输出  $Y_N(\nu-N+1)$  和  $U_{N-1}(\nu-N+1)$  送入  $NN_\Phi$ , 即可得任一时刻  $\nu$  的状态估值  $\hat{X}(\nu)$ 。由于  $X(\nu)$  一般而言反映了实际系统内部各环节的状态, 较难进行现场在线实时测量。通过用  $NN_\Phi$  实现的状态估计器就能更准确、更便捷地估计系统内部状态, 从而得以实现系统的实时控制（见下文）、产品质量控制、生产过程参数提取以及故障早期诊断等一系列重要任务。

### 2. 14. 3 MLFN 在非线性动力系统辨识中的应用

对于一个线性 SISO 系统, 可以用一个线性 ARMA 模型描述其输入和输出之间的关系。具体对于一个  $N$  阶系统, 系统方程如下:

$$y(\nu+1) = \sum_{i=0}^{N-1} \alpha_i y(\nu-i) + \sum_{j=0}^{N-1} \beta_j u(\nu-j) \tag{2-153}$$

对于非线性 SISO 系统, 可以模仿线性系统, 用下列方程描述其输入和输出之间的关系:

$$y(\nu+1) = \Psi[y(\nu), y(\nu-1), \dots, y(\nu-l+1), u(\nu), u(\nu-1), \dots, u(\nu-l+1)]$$

其中  $\Psi[\cdot]$  是一个非线性函数,  $l$  是某个正整数。为简洁起见, 此式可表示为

$$y(\nu+1) = \Psi[Y_l(\nu-l+1), U_l(\nu-l+1)] \quad (2-154)$$

其中  $Y_l(\nu-l+1) = [y(\nu-l+1), y(\nu-1), \dots, y(\nu)]$  以及  $U_l(\nu-l+1) = [u(\nu-l+1), u(\nu-1), \dots, u(\nu)]$ 。如果此非线性系统是围绕原点局部强可观的, 这表明该系统局部线性化后的阶数应等于系统状态向量  $X$  的维数  $N$ 。借助于式 (2-151), 易于导出, 式 (2-154) 应具有下列形式 (注意, 下式中  $Y$  和  $U$  的下标  $N$  表示它们是  $N$  维向量):

$$y(\nu+1) = \Psi[Y_N(\nu-N+1), U_N(\nu-N+1)] \quad (2-155)$$

此映射关系可以用一个神经网络  $NN_\Psi$  予以近似实现, 其结构如图 2-34 所示 其映射特性可表为

$$\hat{y}(\nu+1) = NN_\Psi[Y_N(\nu-N+1), U_N(\nu-N+1)]$$

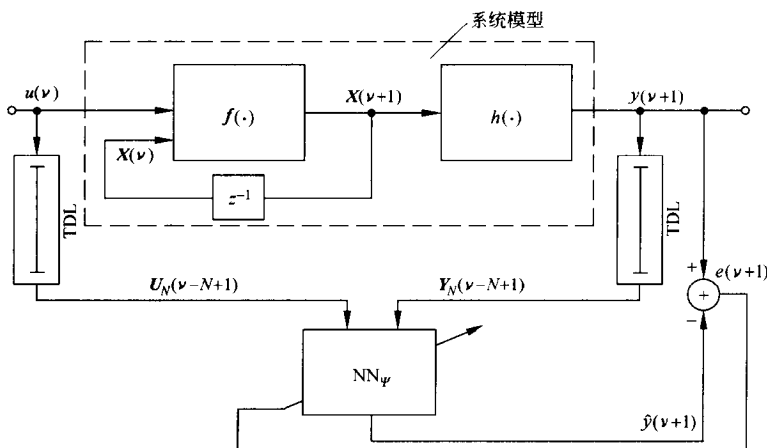


图 2-34  $NN_\Psi$  网络的结构图

$\hat{y}(\nu+1)$  表示  $y(\nu+1)$  的估值。误差  $e(\nu+1) = y(\nu+1) - \hat{y}(\nu+1)$ 。网络经过训练后 将使误差的均方值达到最小。网络结构、训练集采集和训练方法与前一小节所述的常规 MLFN 一致。当训练完成后, 网络进入实际应用时可分成以下两种方式。

### 1. 一步预测方式

用  $y(\nu), \dots, y(\nu-N+1)$  和  $u(\nu), \dots, u(\nu-N+1)$  给出输出估计  $\hat{y}(\nu+1)$ 。这相当于前述 2.13.1 中 (3) 讨论的第一种情况。当  $\nu+1$  时刻以前系统的各输出  $y(\nu), y(\nu-1), \dots$  可以通过现场测试获取时, 可按此方式工作。

### 2. 多步预测方式

若  $\nu+1$  时刻以前, 系统的各输出  $y(\nu), y(\nu-1), \dots$  不能获取, 则只能用  $\hat{y}(\nu), \hat{y}(\nu-1), \dots$  来替代, 其运作方式如图 2-35 所示。

这相当于 2.13.1 小节中 (3) 讨论的第二种情况。这时系统按静态映射方式训练, 而按递归 (有反馈) 方式工作。当  $NN_\Psi[\cdot]$  给出的  $\hat{y}(\nu+1)$  与  $y(\nu+1)$  误差较小时, 才能得到满意的结果。



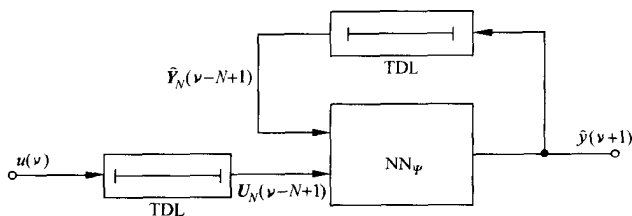


图 2-35  $NN_{\Psi}$  的多步预测方式示意图

## 2.14.4 MLFN 在非线性动力系统调节中的应用

系统调节的目的是，在各种干扰的作用下，使系统的状态在一个所需的平衡点附近保持稳定。如果在系统模型估计得不十分准确且运行时存在干扰的条件下，只要系统的初始状态处于平衡点的某个邻域中，系统状态将稳定地收敛到该平衡点，则称该系统为干扰下强稳定的。对于式 2-146) 描述的非线性系统  $\bar{X} = 0, \bar{u} = 0$  总是它的一个平衡点，可以以它作为对象来研究调节问题。这样，调节问题归结为求反馈函数  $u(v) = g[X(v)]$  使得系统在围绕原点的一个邻域中是干扰下强稳定的。

将上列反馈函数代入式 2-146) 得到

$$X(v+1) = f(X(v), g[X(v)]) = \bar{f}(X(v)) \quad (2-156)$$

设  $X_0$  是围绕原点的某个邻域中的一个状态，若对于任意  $X_0$  皆满足下列条件：

$$\lim_{v \rightarrow \infty} \bar{f}^v(X_0) = 0$$

则对于该邻域系统是强稳定的。函数  $g[\cdot]$  的选择除了使系统成为强稳定的以外，还应使  $X_0$  能够出现的邻域尽可能大。文献[118] 给出了系统在干扰下强稳定的条件，此处不详述。本节只介绍如何求函数  $g[\cdot]$  并如何用神经网络予以实现。先讨论局部线性化系统  $\zeta_L$  的调节问题，再讨论直接对非线性系统  $\zeta$  进行调节的问题。

### 1. 用局部线性化方法解决非线性系统的调节

非线性系统稳定化的最简单方案是采用线性反馈控制。设  $\zeta_L$  是  $\zeta$  围绕原点这个平衡点  $X = 0$  的局部线性化系统 (式 2-151))。文献[120] 证明 如果  $\zeta$  是一个非时变的非线性系统  $X = 0, u = 0$  是它的一个平衡点，若  $\zeta$  围绕  $(0,0)$  线性化后得到的  $\zeta_L$  是可控的，那么  $\zeta$  是局部可控的。文献[120] 还证明，存在一个线性反馈函数  $u = KX^T$  使得  $\zeta$  成为局部渐近稳定的。其中  $K$  是一个  $N$  维行向量，其各分量取实常数值。这样，系统的运行方程是

$$X(v+1) = f(X(v), X(v)K^T)$$

将此反馈函数施行于  $\zeta_L$  (式 2-151)) 得到

$$\delta X^T(v+1) = (A + b^T K) \delta X^T(v)$$

易于证明，如果矩阵  $(\hat{A} + \hat{b}^T K)$  的特征值皆小于 1 那么  $\zeta_L$  是渐近稳定的。如果已经知道了系统  $\zeta$  的函数  $f(\cdot)$  则  $\hat{A}, \hat{b}$  可求，在此基础上，可以求得满足上列特征值要求的  $K$  从而实现系统的稳定化。虽然线性反馈闭环控制确实可以使一个非线性系统围绕平衡点稳定化，

而且构成十分简单（无需使用神经网络），但是其稳定范围比较狭小。采用非线性反馈闭环控制可以有效地扩大稳定范围。

## 2. 用非线性反馈控制器解决非线性系统的调节问题

下面仍以原点这一平衡点为例来进行讨论。构成一个非线性反馈控制闭环系统的理论根据如下所列。第一 设  $\zeta$  围绕原点线性化后所得系统为  $\zeta_L$ 。如果  $\zeta_L$  是可控的 则存在一围绕原点的邻域  $\Omega$ ，当采取某种反馈函数  $u(\nu) = g[X(\nu)]$  时，系统从任一  $X_0 \in \Omega$  出发 经过  $N$  步运行后，系统状态将达到原点。如果采用式（2-156）的表示方法，这一运行过程可以表示为

$$f^N(X_0) = 0, \quad X_0 \in \Omega$$

其证明见文献[118]。注意  $N$  是  $X$  的维数。

第二，存在一个较  $\Omega$  大的围绕原点的邻域  $\mathcal{D}$ （即  $\Omega \subset \mathcal{D}$ ），当  $X_0 \in \mathcal{D}$  时，下列不等式成立：

$$\|f^N(X_0) - 0\| < \|X_0\|, \quad X_0 \in \mathcal{D}$$

这是一个压缩映射，即经过  $N$  步运行后，系统状态虽然不能立即收敛到原点，但是比出发点更靠近原点。这样，只要经几次  $N$  步运行，系统状态总会达到原点。还可以证明， $\mathcal{D}$  总不会小于采用线性反馈控制器时系统的稳定范围。此证明亦见文献[118]。

如上述，设计一个非线性反馈控制系统，关键在于求反馈函数  $u = g(X)$  使得围绕原点的稳定邻域  $\Omega$  和  $\mathcal{D}$  尽可能大。为了用神经网络来实现这一函数，在对其训练时需用一个神经网络  $NN_f$  来模拟系统函数  $f(\cdot)$  的映射功能。此外，在实际现场运行时，状态  $X$  难以实时测量，需要用一个神经网络  $NN_\phi$  对其进行估计。这样，有 3 个神经网络被涉及到，这就是实现函数  $g(\cdot)$  的  $NN_g$ ，实现函数  $f(\cdot)$  的  $NN_f$  和实现状态估计的  $NN_\phi$ 。 $NN_f$  只用于训练，而  $NN_g$  和  $NN_\phi$  用于对实际系统的运作。系统结构如图 2-36 所示。

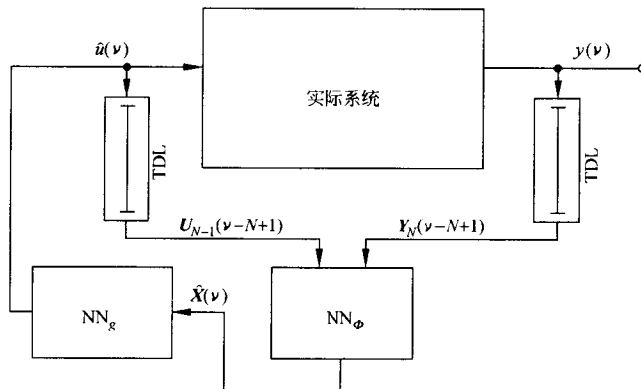


图 2-36 实现非线性反馈控制系统的神经网络结构框图

## 3. 用 $NN_f$ 实现函数 $f(\cdot)$ 的映射功能

在大多数情况下，一个系统  $\zeta$  的状态方程（式 2-144）中的函数  $f(\cdot)$  是未知的。为此可以用一个神经网络  $NN_f$  来逼近此函数的映射关系。网络的映射函数如下：

$$\mathbf{X}(\nu+1) = \text{NN}_f(\mathbf{X}(\nu), u(\nu), \xi)$$

$\hat{\mathbf{X}}(\nu+1)$  表示对  $\mathbf{X}(\nu+1)$  的估计,  $\xi$  是网络的权向量。网络的训练框图如图 2-37 所示 其中的  $u(\nu), \mathbf{X}(\nu), \mathbf{X}(\nu+1)$  皆采自实际系统。网络通过训练求得最佳  $\xi$ , 使得  $\mathbf{X}(\nu+1)$  与  $\hat{\mathbf{X}}(\nu+1)$  之间的误差  $e(\nu+1)$  的均方值尽量小。网络的结构和训练方法与常规 MLFN 无异。

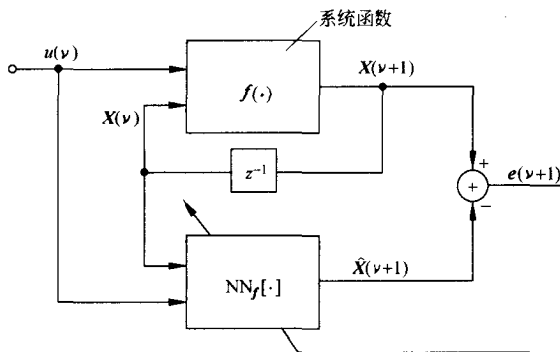


图 2-37  $\text{NN}_f$  网络的训练框图

#### 4. 用 $\text{NN}_g$ 实现反馈函数 $g(\cdot)$

如果设计目标是针对某个具体系统  $\zeta$  求反馈函数  $g(\cdot)$ , 使得系统在一个尽可能大的围绕原点的邻域  $\mathcal{D}$  中达到稳定。从理论上已知, 当系统初始状态  $\mathbf{X}(0) \in \Omega, \Omega \subset \mathcal{D}$  时, 可以进行  $N$  步达到稳定, 即  $\mathbf{X}(N) = \mathbf{0}$ 。当  $\mathbf{X}(0) \in \mathcal{D}$  时 虽然不能达到  $N$  步稳定, 也能实现  $N$  步压缩映射, 即  $\|\mathbf{X}(N) - \mathbf{0}\| < \|\mathbf{X}(0) - \mathbf{0}\|$ 。基于此, 可以用一个神经网络  $\text{NN}_g$  实现映射  $u(\nu) = \text{NN}_g(\mathbf{X}(\nu))$ , 用它来逼近所需的反馈函数  $u(\nu) = g(\mathbf{X}(\nu))$ 。这样 系统具有下列状态转移特性:

$$\mathbf{X}(\nu+1) = \text{NN}_f(\mathbf{X}(\nu), \text{NN}_g(\mathbf{X}(\nu)))$$

其中  $\text{NN}_f(\cdot)$  已经通过 3 所述的方法估计出来。 $\text{NN}_g(\cdot)$  是有待通过训练求得的。为此可以采用图 2-38 所示的框图进行  $\text{NN}_g$  的训练。图中的结构模拟了真实系统经过  $N$  步运行由  $\mathbf{X}(0)$  到  $\mathbf{X}(N)$  的过程。误差  $e(N)$  的定义取决于  $N$  步稳定域  $\Omega$  和  $N$  步压缩映射域  $\mathcal{D}$  的范围, 而此二者不可能在训练之前就确定的。为此可以采取下述的策略。首先用两个参数

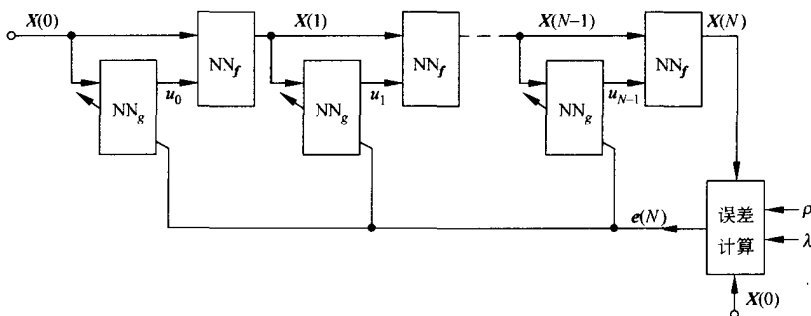


图 2-38  $\text{NN}_g$  的训练框图

$\rho$  和  $\lambda$  来定义这两个域 ( $0 < \rho, 0 < \lambda < 1$ ),

① 若  $\|X\| < \rho$  则  $X \in \Omega$ ;

若  $X(0) = X, \|X(N)\| \leq \lambda \|X(0)\|$  则

$$X \in \mathcal{D}$$

网络学习策略是对网络进行训练,使  $e(N)$  模的均方值达到最小的同时,令  $\rho$  和  $\lambda$  也发生变化。 $\rho$  的变化规律是在学习开始阶段,选较小  $\rho$  值然后逐渐变大。 $\lambda$  的变化规律是  $\lambda$  先选择接近于 1 的值,然后逐渐缩小。 $e(N)$  按下式计算:

$$e(N) = \begin{cases} X(n), & \|X(0)\| < \rho \text{ 或 } \|X(N)\| > \lambda \|X(0)\| \\ 0, & \text{其他情况} \end{cases}$$

由于一开始时网络参数尚未调整得当,  $\rho$  选得小一些 (即  $\Omega$  较小), 易于实现  $N$  步稳定化。如果所选的  $\rho$  值在学习开始阶段仍不能使得系统大部分达到  $N$  步稳定, 则应选更小的  $\rho$  值进行训练; 如果已全部达到  $N$  步稳定, 则应令  $\rho$  加大后再学习; 此后, 随着网络参数的调整,  $\rho$  逐渐加大, 即  $N$  步稳定范围  $\Omega$  逐渐扩大。而开始  $\lambda$  选得接近于 1 (即压缩得很少) 是为了使  $\mathcal{D}$  尽量大。随着网络参数的调整,  $\lambda$  可逐渐缩小, 因为前者可以补偿后者, 使得  $\mathcal{D}$  保持不变或扩大的同时, 压缩比进一步改善, 从而加快系统收敛到稳定点的速度。

图 2-38 所示的结构, 仍是一实现静态映射的 MLFN, 所以可采用静态映射的学习算法来解决一个动态反馈的问题。这是此方案的一个突出优点。

### 5. 用 $X(\nu)$ 替代 $X(\nu)$

在求得了  $NN_k$  后, 即由一个实际系统的状态  $X(\nu)$  求得反馈控制输入  $u(\nu) = NN_k[X(\nu)]$ 。由于实际测量的困难, 在实际操作中总是用  $NN_k$  得到的状态估值  $\hat{X}(\nu)$  替代  $X(\nu)$ 。因此反馈输入信号也是  $u(\nu)$  的估值  $\hat{u}(\nu) = NN_k[\hat{X}(\nu)]$  (见图 2-36)。

## 2.14.5 MLFN 在非线性动力系统跟踪中的应用

### 1. 跟踪功能

设  $y^*(\nu)$  是一个希望实现的目标输出序列, 设其为均匀有界的。对于一个具体系统, 实现跟踪功能在于求一个输入序列  $u(\nu)$  使得系统的实际输出  $y(\nu)$  与理想输出  $y^*(\nu)$  之间的误差随着  $\nu$  的增大而趋向于 0。这可以表示为

$$\lim |y(\nu) - y^*(\nu)| = 0$$

### 2. 线性系统的跟踪功能

设一个线性系统在时刻  $\nu$  的状态为  $X(\nu)$  在  $\nu$  时刻及其以后时刻的系统输入为  $u(\nu)$ ,  $u(\nu-1), \dots$  则系统在  $\nu+\mu$  时刻的输出  $y(\nu+\mu)$  由下式

$$y(\nu+\mu) = CA^\mu X(\nu) + CA^{\mu-1} b^T u(\nu) + \dots + Cb^T u(\nu+\mu-1)$$

决定 (据式 2-147))。如果

$$Cb^T = CAB^T = \dots = CA^{d-2}b^T = 0, \quad CA^{d-1}b^T \neq 0$$

$d$  是一个不小于 1 的整数。则当  $\mu = d$  时可得

$$y(\nu+d) = CA^d X(\nu) + CA^{d-1} b^T u(\nu) \quad (2-157)$$

此式表明,  $\nu$  时刻的输入在  $\nu+d$  时刻才对输出造成影响。因此  $d$  表示输入至输出的系统延

时，常称之为系统的相对阶。

设  $\nu + d$  时刻的系统理想输出为  $y^*(\nu + d)$ ，则由式 (2-157) 可求得系统在  $\nu$  时刻的输入  $u(\nu)$  为

$$u(\nu) = \frac{y^*(\nu + d) - \mathbf{CA}^d \mathbf{X}^T(\nu)}{\mathbf{CA}^{d-1} \mathbf{b}^T}$$

如将此  $u(\nu)$  输入系统，则求得各离散时刻  $\nu + 1$  的系统状态  $\mathbf{X}(\nu + 1)$  是(据式 (2-147))

$$\mathbf{X}^T(\nu + 1) = \underbrace{\left[ \mathbf{A} - \frac{\mathbf{b}^T \mathbf{CA}^d}{\mathbf{CA}^{d-1} \mathbf{b}^T} \right]}_{\mathbf{B}} \mathbf{X}^T(\nu) + \frac{\mathbf{b}^T y^*(\nu + d)}{\mathbf{CA}^{d-1} \mathbf{b}^T}$$

如果矩阵  $\mathbf{B}$  的特征值皆小于 1，那么无论系统初始状态如何，只要  $y^*(\nu + d)$  恒有界， $\mathbf{X}(\nu + 1)$  也恒有界，因而  $u(\nu + 1)$  有界。这样，系统可以依赖有界输入，实现对理想有界输出的跟踪。这种系统称为最小相位系统。若系统是非最小相位的，则做不到这一点。

### 3. 非线性系统的跟踪功能

非线性系统的全局跟踪性能很难分析。为此，下面的分析只局限于原点附近跟踪性能的研究。首先，要进行若干假设和定义，然后给出有关非线性系统原点附近跟踪性能的一些基本定理。

(1) 求非线性系统  $\zeta$  围绕平衡点(原点)的相对阶  $d$ 。按照式 (2-146)，一个非线性系统  $\zeta$  在  $\nu + 1$  时刻的输出  $y(\nu + 1)$  为

$$y(\nu + 1) = h(f(\mathbf{X}(\nu), u(\nu)))$$

此式右侧可以表示为复合函数  $f \circ h(\mathbf{X}(\nu), u(\nu))$ 。设  $\mathbf{X}(\nu)$  和  $u(\nu)$  的定义域为  $\chi$  和  $\mathcal{U}$ 。若存在一个围绕原点的域  $\Omega$ ， $\Omega \subset \chi \times \mathcal{U}$ ，使下式成立：

$$\left. \frac{\partial^k [f \circ h(\cdot)]}{\partial u^k} \right|_{\mathbf{x}, u \in \Omega} = 0, \quad k = 1, 2, \dots, d-1,$$

$$\left. \frac{\partial^d [f \circ h(\cdot)]}{\partial u^d} \right|_{\mathbf{x}, u \in \Omega} \neq 0$$

其中  $d$  是一个不小于 1 的整数。则称系统  $\zeta$  在原点附近的相对阶(延时)为  $d$ 。但是，对于有些非线性系统，即使  $\Omega$  很小也不存在能够加以明确定义的  $d$ 。为此需要作一个假定：所讨论的系统在  $\mathbf{X}$  的包含原点的一个邻域  $\Omega_X$  中存在能够明确定义的相对阶  $d$ 。这时可以证明<sup>[119]</sup>，必然存在围绕原点的邻域  $\Omega_X$  和  $\Omega_y$ ，当  $\mathbf{X}(\nu) \in \Omega_X, y^*(\nu + d) \in \Omega_y$  时，一定可以求出一个  $u(\nu)$  作为系统的输入(不管  $\nu$  以后系统的输入是什么)，总可以使得  $\nu + d$  时刻的系统实际输出等于理想输出，即  $y(\nu + d) = y^*(\nu + d)$ 。

(2) 如果系统  $\zeta$  能够跟踪某个目标输出序列  $y^*(\nu)$ ，则称  $\zeta$  对  $y^*(\nu)$  具有产生功能。线性最小相位系统对任何有界  $y^*(\nu)$  皆具有产生功能。对于非线性系统，设 ① 中所做的假定成立，且假定存在一个  $u(\nu)$  的集合  $\Omega_u$  和一个  $\mathbf{X}(0)$  的集合  $\Omega_{X_0}$ ， $\Omega_{X_0} \subset \Omega_X$ ，当  $\mathbf{X}(0) \in \Omega_{X_0}$  及  $u(\nu) \in \Omega_u, \forall \nu, \mathbf{X}(\nu) \in \Omega_X, \forall \nu > 0$ 。这时可以证明下列定理成立<sup>[119]</sup>：存在一围绕原点的  $y(\nu)$  的邻域  $\Omega_y$ ，对于任何目标输出序列  $y^*(\nu) \in \Omega_y$ ，当  $\nu \geq d$  时， $\zeta$  对该序列具有产生功能。

### 4. 用神经网络构成非线性系统跟踪控制器

(1) 用一个神经网络  $NN_d$  来逼近由  $\mathbf{X}(\nu), u(\nu)$  产生  $y(\nu + d)$  的映射。 $NN_d$  的训练框

图如图 2-39 所示。图的上半部模拟一个非线性系统  $\zeta$  由  $X(\nu)$ ,  $u(\nu)$  产生  $y(\nu+d)$  的过程。其中  $NN_f$  逼近函数  $f(\cdot)$  的映射功能, 其训练方法已在上述 2.14.4 小节 3 中阐明。 $NN_h$  也是一个神经网络, 用来逼近函数  $h(\cdot)$  的映射功能, 其训练方法与  $NN_g$  相似, 不赘述。图的下半部用  $NN_d$  直接逼近由  $X(\nu)$ ,  $u(\nu)$  产生  $y(\nu+d)$  的映射。由  $NN_d$  产生  $y(\nu+d)$  的估值  $\hat{y}(\nu+d)$ 。通过调整  $NN_d$  的权, 使此二者的误差  $e(\nu+d)$  的均方值达到最小。采用  $NN_d$  可以使计算大为简化, 以便于控制器在训练中使用。

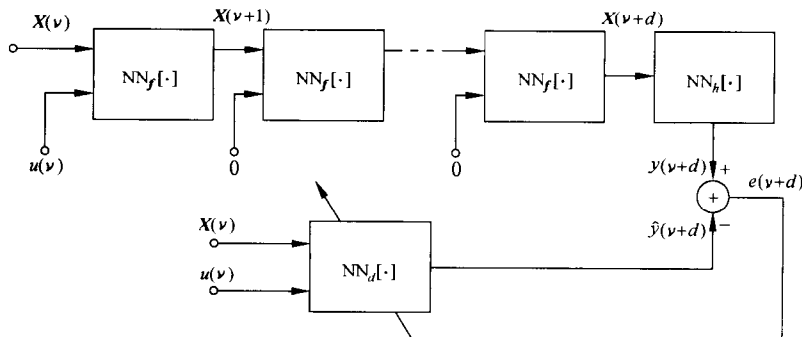


图 2-39  $NN_d$  的训练框图

(2) 用一个神经网络  $NN_c$  来实现系统跟踪的反馈控制器。设时刻  $\nu$  的系统状态为  $X(\nu)$  若要求系统在时刻  $\nu+d$  达到目标输出  $y^*(\nu+d)$ , 那么该神经网络的作用是由  $X(\nu)$ ,  $y^*(\nu+d)$  产生所需的反馈控制信号 (即系统输入)  $u(\nu)$ , 使得系统的真实输出  $y(\nu+d)$  与  $y^*(\nu+d)$  之间的误差  $e(\nu+d)$  的均方值达到最小。 $NN_c$  的训练框图如图 2-40 所示。其中  $NN_d$  由  $X(\nu)$ ,  $u(\nu)$  产生系统在  $\nu+d$  时刻输出的估值  $\hat{y}(\nu+d)$ 。 $u(\nu)$  由  $NN_c$  产生,  $u(\nu) = NN_c[X(\nu), y^*(\nu+d)]$ 。通过训练, 调整  $NN_c$  的权参数, 使得  $e(\nu+d)$  的均方值达到最小。 $NN_d[\cdot]$  可称为辨识网络,  $NN_c[\cdot]$  可称为控制网络, 可与 2.13.6 小节的图 2-31 对比。

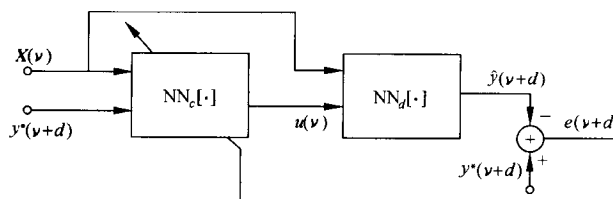


图 2-40  $NN_c$  训练框图

## 2.15 MLFN 应用举例之二 —— 手写数字识别

MLFN 在很多应用领域中已进入实用化阶段。许多成果的取得经过了漫长的研究和开发过程, 其中进行了细致的推敲、改进和反复的测试, 特别在选择网络结构、提取输入信号特征和建立训练、测试数据库等方面尤需注意。手写体阿拉伯数字 0~9 的自动识别是

MLFN 获得成功应用的一个典型实例，它既具有很大的实用价值（如信函自动分检），又具有理论意义（因为此问题已为多人研究过，所以新算法或新方案都可以用它来检验其效果）。此外，它也可以成为其他各种手写符号和文字识别的基础。这就是包括 AT&T 在内的许多大公司和研究所投入巨资研究这个问题的原因。

## 2.15.1 衡量一个手写体阿拉伯数字自动识别系统性能的指标,训练和测试数据库

一个识别系统的首要指标是误识率。误识的代价往往很大，例如信函的邮编被误识后将错送到千里以外。为了减少误识造成的损失，可以对一些判别可信度不高的被识别对象拒识（由拒识筛选下来者可用其他方法处理，例如在信函分检系统中进行人工分检）。这样，可以用误识率达到一定标准（如 1%）的条件下，拒识率的高低来衡量一个识别系的优劣；也可以将此二者互换。

误识率的高低除了取决于识别系统本身性能外，还与训练集和测试集的容量和特性相关。例如,AT&T 公司用了两种数据库。一种是美国邮政服务数据库（U. S. postal service database）它的训练集包含 7291 个手写数字，测试集包含 2007 个。这些数字是从邮局的死信上复制下来的，书写不受约束且由风格各异、仔细程度不一的许多人写的。这些字的尺寸不同、笔划粗细不同、使用工具不同且其中颇多含糊之处。有些数字串书写时连在一起，剪开后模糊更甚。总之，这是一个识别难度颇大的数据库，即使由人目验，其误识率也达到 2.5%。另一种是由美国标准与技术国家研究所（U. S. National Institute of Standards and Technology）提供的 USNIST 数据库。它们以两张 CD ROM 的形式提供，其中包含约 60000 个训练样本和 10000 个测试样本。训练集中的数字由一批美国人口调查工作人员书写（付酬）；测试集中的数字由大量高校学生书写，他们的合作态度不如前者，字迹较草率。因而此数据库的训练集和测试集中数字的特征分布不尽一致。但是，这个库的识别难度较低。除了误识率外，一个数字识别系统的指标还包括训练时间、识别时间以及系统的存储容量等。

## 2.15.2 Le Net 1<sup>[96,97]</sup>

Le Net 1 是 20 世纪 80 年代末由 Le Cun 领导的一批科学家开发的一个基于神经网络的手写数字自动识别系统。它的主体部分是一个 5 层 MLFN，它的功能既包括特征提取又包括模式分类，网络的训练采用标准 BP 学习算法。下面首先介绍预处理部分，再介绍网络本身。

预处理部分所作的是首先将大小不一的字体通过线性变换规格化为统一的、具有  $16 \times 16$  个像素的图像。线性变换的结果是使原来只有两个灰度级（白与黑）的像素具有多个灰度级，因此需将其值规格化到  $[-1, 1]$  范围内。然后，在此图像周围加一个纯白（灰度级等于 1）的边框 形成一个  $28 \times 28$  像素的图像（边框的宽度是 5 个像素）加边框的作用是避免边缘的不连续效应。

Le Net 1 的主体是一个 5 层 MLFN 其结构如图 2-41 所示。它的输入是预处理部分

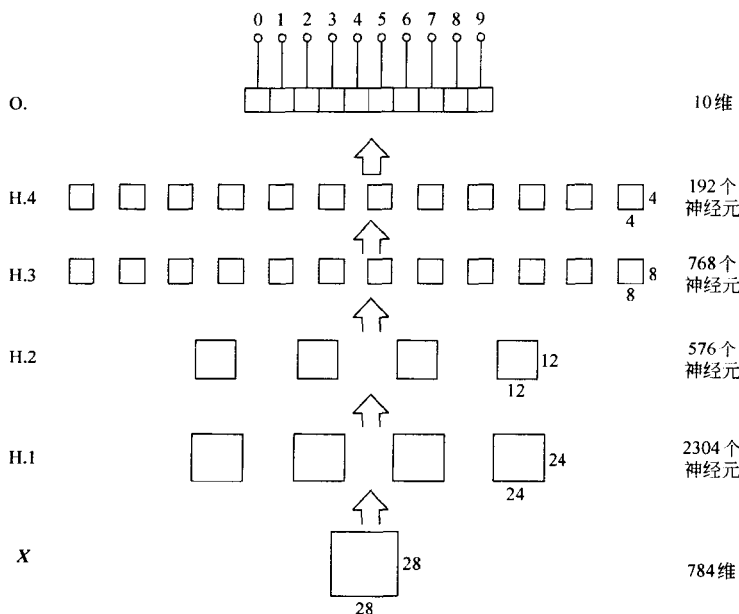


图 2-41 Le Net 1 的结构图

形成的  $28 \times 28$  像素图像，可将其表示为一个 784 维向量  $\mathbf{x}$  每个分量取值范围为  $[-1, 1]$ 。网络由 5 层构成，它的 4 个隐层分别用 H.1 ~ H.4 表示，输出层用 O. 表示。

第一隐层 H.1 由 4 块神经元阵列构成，分别用 H.1.1 ~ H.1.4 表示。其中每个阵列包含  $24 \times 24$  个神经元，所以 H.1 共包含 2304 个神经元 ( $4 \times 24 \times 24$ )。这 4 个阵列与  $\mathbf{x}$  的连接方式相同。在一个阵列中，每个神经元只与  $\mathbf{x}$  的一个  $(5 \times 5)$  像素局部图像块相连接：同一阵列中横向相邻或纵向相邻的两个神经元所连接的  $\mathbf{x}$  局部图像块沿横向或纵向移动一个像素。H.1 中每个神经元共  $5 \times 5$  个输入权，加上一个阈值参数，共 26 个参数。同一阵列中各神经元取相同参数，不同阵列取不同参数。这样，H.1 总共有 104 个可调的自由参数 ( $4 \times 26$ )。这里所取的正是局部连接和权分享方案。H.1.1 ~ H.1.4 分别检测图像中的一个局部特征，它们是：短横线段、短竖线段、左下至右上短斜线段和左上至右下短斜线段。这样，由输入层  $\mathbf{x}$  至 H.1 的全部权数（包括阈值参数）为 59904 ( $26 \times 2304$ )，而自由调节参数只有 104 个。由于 H.1 中每个神经元只涉及输入图像的一个局部小区，所以也是一种局部接收场的方法。

第二隐层 H.2 也由 4 块神经元阵列构成，它们是 H.2.1 ~ H.2.4 分别与 H.1.1 ~ H.1.4 一一对应。H.2 的每个阵列由  $12 \times 12$  个神经元排列而成。H.1 的每个阵列划分成  $12 \times 12$  个互不重叠的小区，每个小区含  $2 \times 2$  个神经元。这 4 个神经元的输出乘以相同的权后相加，再加上一个阈值参数后，送 H.2 中相应阵列的对应神经元。由 H.1.k 至 H.2.k ( $k = 1 \sim 4$ ) 的权及阈值对同一阵列取相同值，不同阵列可取值不同。这样，由 H.1 至 H.2 的权及阈值参数共有 2880 个 ( $2304 + 576$ )，而可调节的自由参数只有 8 个 ( $2 \times 4$ )。H.2 的作用与 H.1 不同，不是实现特征提取而是进行取平均以及降低采样率。

第三隐层 H.3 由 12 块神经元阵列 H.3.1 ~ H.3.12 构成，每个阵列由  $8 \times 8$  个神经



元排列而成，其作用类似于 H. 1，从低一级的特征提取高一级的复合特征。H. 3 中每一个神经元阵列中的任一神经元只与 H. 2 中某一个阵列或某二个阵列的同位置 ( $5 \times 5$ ) 局部区域神经元连接且取相同权。其中的 4 个阵列 (H. 3. 1 ~ H. 3. 4) 的任一神经元只与 H. 2. 1 ~ H. 2. 4 对应位置的 25 个神经元相连，共 25 个权再加 1 个阈值，每个阵列有 26 个自由参数。其他 8 个阵列 (H. 3. 5 ~ H. 3. 12) 的任一神经元与 H. 2. 1 ~ H. 2. 4 中的两块阵列 (包括一块阵列用两次) 对应位置的 50 个神经元相连，共 50 个权再加一个阈值，每个阵列有 51 个自由参数。这样由 H. 2 至 H. 3 共有 512 个自由参数 ( $4 \times 26 + 8 \times 51$ )，而权及阈值的总数是 32768 个 ( $4 \times 64 \times 26 + 8 \times 64 \times 51$ )。

第四隐层 H. 4 也由 12 块神经元阵列 H. 4. 1 ~ H. 4. 12 构成，分别与 H. 3. 1 ~ H. 3. 12 一一对应，其作用类似于 H. 2，实现取平均和降低采样率。H. 4 的每块阵列由  $4 \times 4$  个神经元排列而成，H. 3 的每块阵列划分成  $4 \times 4$  个互不重叠的小区，每个小区含  $2 \times 2$  个神经元，其输出乘相同权后与阈值相加，再送至 H. 4 相应阵列对应位置的神经元；同一阵列取相同权和阈值，不同阵列可取不同权和阈值。这样，H. 3 至 H. 4 的可调自由参数共 24 个 ( $2 \times 12$ )，而权和阈值的总数为 960 个 ( $768 + 192$ )。

网络的输出层 O. 共包含 10 个神经元，每个神经元与一个数字对应。每个神经元与 H. 4 的 192 个神经元全部连接，所以有 192 个输入权和一个阈值参数。H. 4 至 O. 的权总数和自由参数都是 1930 个 ( $193 \times 10$ )。网络中全部神经元皆取 tanh 函数型 Sigmoid 函数，其输出值变化范围是  $-1 \sim 1$ 。当输入向量  $X$  属于数字  $i$  时，O. 的第  $i$  端输出应等于 1，其他各端输出应等于  $-1$ 。在对网络进行训练时，网络的实际输出与上列理想输出的均方误差应达到最小 (对训练集取平均)。网络的学习采用 BP 算法。

归纳起来，网络的 4 个隐层起特征提取作用，最后形成 192 维特征向量 (H. 4 的输出) 由网络的输出层根据此特征向量完成分类功能。网络中共包含 3850 个神经元，98442 个权和阈值 ( $59904 + 2880 + 32768 + 960 + 1930$ )，其中独立调节的自由参数 2578 个 ( $104 + 8 + 512 + 24 + 1930$ )。

Le Net 1 对美国邮政服务数据库的实验结果如下。整体误识率达到 5.1% (用训练集训练，测试集测试)。如果允许拒识，则误识率随拒识率的增加而下降的曲线如图 2-42 所示。为了达到 1% 的误识率，拒识率是 9.6%。拒识的准则是比较 O. 最大输出端和次大输

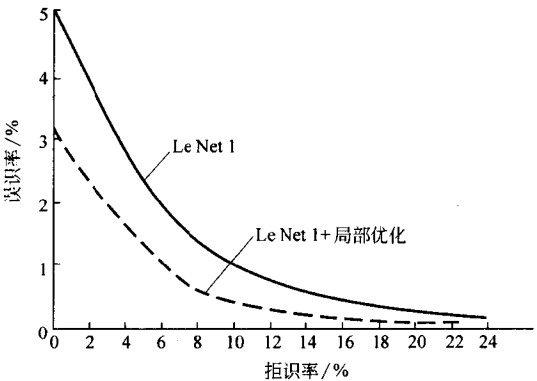


图 2-42 误识率与拒识率的关系图

出端，二者输出值之差异如果小于某一门限值，则予以拒识。对此数据库还做了另一项实验 将原包含 7291 个手写数字的训练集扩充为再包含 2549 个印刷体数字，将原包含 2007 个手写数字的测试集扩充增加 700 个印刷体数字。按此方案进行 30 次 BP 学习后，训练集内的输出均方误差 (MSE) 为 0.017 误识率为 1.1% 测试集的输出 MSE 为 0.024 误识率为 3.4 (印刷体数字一个不错) 训练和测试均用一台 Sun Sparc 工作站施行 (加装一个 SN2 模拟器) 全部运行时间为 3 天。

Le Net 1 对 USNIST 数据库的实验结果如下。这个数据库的容量较上一个扩大了很多倍且更加规范化。但是它的训练集和测试集的特性有较大差异，如果直接使用它们，既不符合实际使用情况，实验结果也差强人意。因此在 Le Net 1 的实验中 将 USNIST 的训练集和测试集各自分成两部分，再各取一半构成新的训练集和测试集，每个集中包含约 4 万个手写数字。Le Net 1 经训练集训练后，测试集的误识率为 1.7%。由于此数据库的样本较规范，所以效果明显优于邮政数据库。

### 2.15.3 Le Net 4<sup>[98]</sup>

Le Net 4 是 Le Net 1 的扩展形式，它的各个隐层包含更多块神经元阵列，从而能够搜索输入信号中的更多特征。此外 在 H. 4 和 O. 之间再增加一个隐层 H. 5, H. 5 与 O. 配合完成分类功能。Le Net 4 共有 17 000 个自由参数 (Le Net 1 只有 2578 个) 实现一次识别需进行 260000 次乘 / 加运算 (Le Net 1 是 140000 次)。Le Net 4 对 USNIST 数据库的实验结果是 (与 Le Net 1 取相同实验方案) 测试集误识率达到 1.1%。这个结果相当出色。

### 2.15.4 Le Net 1 + 局部优化<sup>[99]</sup>

在 Le Net 1 的基础上采用局部优化法还可以明显改善系统的识别效果。实验针对美国邮政服务数据库进行。第一阶段的训练仍按照上述 2.15.2 的步骤实施，此阶段结束后 将输入  $X$  到 H. 4 的各个权及阈值固定下来。第二阶段以 H. 4 输出的 192 维特征向量作为分类器的输入，针对 H. 4 至 O. 的 1920 个参数实施局部优化方案。具体作法如下。每输入一个待识别的测试集向量  $X_i$ ，就在训练集中搜索  $K$  个向量  $X_{ij}, j = 1 \sim K$  后者的 192 维特征向量与前者的 192 维特征向量之间的欧氏距离较训练集其他成员与  $X_i$  的特征向量之间的欧氏距离小。然后用这  $K$  个向量按 BP 算法对 H. 4 至 O. 之间的 1920 个参数进行训练，使得相应的均方误差达到最小。在学习过程中采用了权衰减法 (见 2.11.2 小节中 (1))。实验证明，取  $K = 200$  及规格化系数  $\gamma = 0.01$  时效果最好。此方案的实验结果是测试集误识率达到了 3.3%。注意，人对于此测试集的目标误识率为 2.5%，二者相比可见此方案的明显优越性。这一系统的误识率相对于拒识率的变化曲线也在图 2-39 中示出 在达到 1% 误识率时的拒识率为 6.2%。可看到，采用局部优化后的性能明显超出原 Le Net 1 系统。实际上，当拒识率为 17% 时，全部测试集的测试结果中只产生了一个“错误”，而这个“错误”是原数据库加标记时标注错误。这个方案需要将全部训练集存储起来，每识别一次都要搜索  $K$  个训练样本且进行局部优化参数学习，这使系统的存储量和计算开销很大，以致给实用化造成困难。此方案对 USNIST 数据库也进行了实验，由于原系统指标本来就

已很高，所以改进不很明显。

### 2.15.5 多网络法<sup>[99,100]</sup>

按照多网络的原理，在 Le Net 4 的基础上构成了一个新系统，可以称为增强型 (boosted) Le Net 4。整个系统包括 3 个 Le Net 4。第一个 Le Net 4 用全部训练集数据按常规方法进行训练。第二个 Le Net 4 的训练集包含两部分，一部分是第一个 Le Net 4 误识的训练集样本，另一部分是从第一个 Le Net 4 正确识别的训练样本中挑选出来的，二者各占 50%。第三个 Le Net 4 的训练样本由前两个 Le Net 4 识别结果不一致的那些训练集样本构成。实验针对 USNIST 数据库进行，并且对训练集和测试集采取了前述的拆分和重组。由于 Le Net 4 的测试集误识率都达到了 1.1%，训练集误识率更低，因而提供第二个 Le Net 4 用的误识训练样本数相当少。因此第二、三 Le Net 4 的训练样本都不够充分。为了补其不足，将训练集中的样本进行变形后补充到训练集中。变形方法包括仿射 (affine) 变换和线条粗细的变化。当三个 Le Net 4 分别训练完成后，每输入一个待识别对象 (取自测试集) 就将其送到三个网络，然后取三个网络输出的算术平均值作为系统的输出。对 USNIST 数据库所做的实验结果是，测试集误识率达到 0.7%。这是迄今为止所有方案中一项最佳的实验结果，是一个水平很高的结果。

### 2.15.6 TDC 网络<sup>[101]</sup>

TDC 是 tangent distance classifier 的缩写。采用此方案的优点是，当一个被识别符号作小范围的位移、旋转、尺度伸缩和线条粗细变化时，识别结果不会有显著变化。此方案在美国邮政服务数据库上所做的实验结果达到了极佳的水平，测试集误识率为 2.7%。对 USNIST 数据库所作实验结果是误识率为 1.1%。

## 2.16 MLFN 应用举例之三 —— 语音识别

语音识别的最终目标是以自然述说的语音为媒体实现人与计算机之间的直接通信，以语音替代键盘实现计算机的信息输入无疑具有极大的应用价值<sup>[32]</sup>。神经网络从其发展的初期即引起了语音识别研究者的高度重视，但是早期的研究偏重于孤立的音素、音节或词的识别课题。多项研究工作证明，采用神经网络后识别效果较之传统方法有明显改进。例如，在 1988 年，A. Waibel 等用局部连接和权分享构成的 MLFN (即 TDNN) 来识别最难区分的 3 个浊塞音：“[b]”，“[d]”，“[g]”，取得了非常好的实验结果<sup>[102]</sup>。通过对比实验证实，采用 TDNN 时，这 3 个音的正确识别率分别为 98.8%、99.1%、97.5%，而在完全相同的条件下，采用隐含马尔可夫模型 (HMM) 的识别系统对这 3 个音的正确识别率是 92.9%、97.2%、90.9%。这一结果当年曾给人深刻的印象。但是孤立语音识别的用途毕竟非常有限，现在和未来的研究和开发重点是大词汇表、非特定人、连续语音识别<sup>[32]</sup>。这一研究课题的困难之处在于，连续语音中的各个词或各个音节之间不存明显的分界线，不可能首

先将它们分割成孤立的词或音节，再分别予以识别。这就是说对连续语音中的各个词或音节的分割和识别只能结合在一起进行，从而使问题复杂化。当前各种成功的连续语音识别系统几乎都是基于 HMM 的<sup>[32]</sup>，尽管这些系统已经取得了长足的进展，但是仍有很大的改进余地。目前，很多研究者都关注于将 ANN 与 HMM 相结合 构成 HMM/ANN 混合系统。所用的 ANN 主要是 MLFN(MLP、RBF 等 )下面将重点介绍 HMM/MLP 混合系统。

### 2.16.1 基于 HMM 的连续语音识别系统简介

HMM 的基本原理介绍如下（详见文献[32]）。可以将一段有开始有结束的连续语音（例如一句完整的话语）看成为一具有层次的结构。它的最底层的构成单位是音子（phone），按照一定的发音规则，可以由音子组成不同的词（word）。再上一层 按照一定的句法（syntax）或文法（grammar），可以由词组成不同的句子（sentence）。

美国的 MIT 建立了一套称为 TIMIT 的语音数据库<sup>[105]</sup> 其中共用了 62 个音子。例如，英语中的短语“the cat”由 7 个音子组成，如图 2-43 所示。其中第一个音子前的空白圆表示前方词的最后一个音子，而最终音子后的空白圆表示后续词的第一个音子。图中“tcl”音子至最终空白圆的跨越音子“t”的弧，表示后者可能被省略。每一种音子表示一种特定的发音模式。

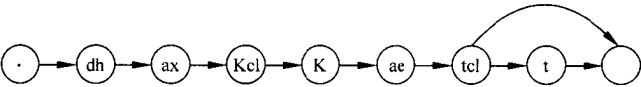


图 2-43 “the cat” 的音子组成

为了描述各种不同的模式，首先应该将一段与一个音子相应的数字化语音序列转换为一个特征向量  $\mathbf{X}$  的序列，记为  $\mathbf{X}_n, n = 1, 2, \dots$ 。下标  $n$  是时序标号，也是一个语音帧的标号。相邻帧的间隔一般为 10ms，帧的长度通常是 20ms ~ 30ms；当语音采样率为  $f_s = 8\text{kHz}$  时 相邻帧间隔为 80 个样点 帧长为 160 ~ 240 个样点。对任何  $n, \mathbf{X}_n$  由相应的帧语音采样序列经过某种变换得到。常用的特征向量举例如下。 $\mathbf{X}_n$  是一个 26 维行向量，其分量包括 12 个听觉加权倒谱系数，12 个此系数的帧间差分，1 个帧能量值，1 个帧能量的差分<sup>[32,103]</sup>。

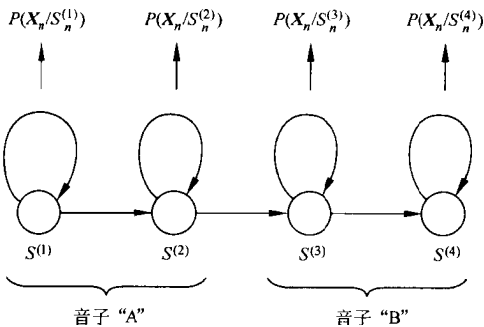


图 2-44 音子的 HMM 表示示意

其次，每一个音子可以用具有若干个状态的 HMM 来表示。在图 2-44 中给出了一个示例，其中音子 A' 包含  $S^{(1)}$ 、 $S^{(2)}$  两个状态，音子 B' 包含  $S^{(3)}$ 、 $S^{(4)}$  两个状态，二者串接成一个包含两音子的 HMM 系统。

接着，用此模型来描述一个特征向量序列的产生过程。若在某一时序  $n$ ，系统处于状态  $S^{(i)}$  这可以记为  $S_n^{(i)}$  若下一时序  $n+1$  系统所处状态为  $S^{(j)}$  则可记为  $S_{n+1}^{(j)}$  等等。在状态转移中，前一状态只能在有箭头指出的几个方向中选择后一状态。例如，图 2-44 模型可以给出下列状态序列（设  $n = 1 \sim 12$ ）：

$S_1^{(1)} S_2^{(1)} S_3^{(1)} S_4^{(2)} S_5^{(2)} S_6^{(2)} S_7^{(2)} S_8^{(3)} S_9^{(3)} S_{10}^{(4)} S_{11}^{(4)} S_{12}^{(4)}$  系统在每一个状态都产生一个特征向量，由上列状态序列产生的特征向量序列可以记为

$$\mathbf{X}_1 \quad \mathbf{X}_2 \quad \mathbf{X}_3 \quad \cdots \quad \mathbf{X}_{11} \quad \mathbf{X}_{12}$$

一个 HMM 的特性用它的状态之间转移概率和某一状态下产生某个特征向量的概率来描述。对于目前普遍采用的一阶马尔可夫链，系统在任一  $n+1$  时序所处的状态  $S_{n+1}^{(j)}$  具有何种发生概率只取决于  $n$  时序下系统已处于何种状态  $S_n^{(i)}$  这可以用条件概率  $P(S_{n+1}^{(j)} / S_n^{(i)})$  来描述。系统若于时序  $n$  处于状态  $S_n^{(i)}$ ，则其产生特征向量  $\mathbf{X}_n$  的概率可以用条件概率  $P(\mathbf{X}_n / S_n^{(i)})$  来描述（见图 2-44）。当一个 HMM 由多个音子串接而成时（如图 2-44 所示情况），可以对所有音子的各个状态予以统一的编号，若一个 HMM 共有  $I$  个状态，则状态的集合可表示为  $\{\mathbf{S}\} = \{S^{(i)}, i = 1 \sim I\}$ （图 2-44 所示例中， $I = 4$ ）。若一段话语共包含  $N$  帧语音，则系统所经历的  $N$  个状态和相应的  $N$  个特征向量可以用两条链  $S_n^{(i_n)}, n = 1 \sim N$  和  $\mathbf{X}_n, n = 1 \sim N$  表示，前者隐于系统内部，是看不见的，可见的只是后者，所以称之为 HMM 系统。

下面用  $\mathbf{X}$  表示一个特征向量链（序列  $\mathbf{X}_1 \mathbf{X}_2 \cdots \mathbf{X}_N$ ）。一段长度为  $N$  帧的话语由各种可能的词串接而成，而后者又由相应的音子构成，音子由状态构成。这样，任何一段话语实质上可能由若干不同的 HMM 产生，每个 HMM 表示一个词串，该词串用相应的状态集  $\mathbf{S}$ 、状态转移概率  $P(S_{n+1}^{(j)} / S_n^{(i)})$  和特征向量产生概率  $P(\mathbf{X}_n / S_n^{(i)})$  描述其特性。各个不同的 HMM 设有  $K$  个，则可以记之为  $\mu^{(k)}, k = 1 \sim K$ ， $\mu^{(k)}$  表示上列各项特性的总和。如果已知各  $\mu^{(k)}$ ，且能够计算后验概率  $P(\mu^{(k)} / \mathbf{X})$ ，那么语音识别任务就在于求  $k^*$ 。

$$k^* = \operatorname{argmax}_k P(\mu^{(k)} / \mathbf{X})$$

并将  $\mu^{(k)}$  所相应的词串作为一段话语  $\mathbf{X}$  的识别结果。如果识别结果与该段话语相应词串一致，识别正确；反之，为误识。语音识别的研究课题包括尽量准确地计算各  $P(\mu^{(k)} / \mathbf{X})$  以及尽量准确地估计各个 HMM 系统  $\mu^{(k)}$  的参数（其目的也是为了更准确地计算前者）。对于每个  $\mu^{(k)}$  有两套参数，这就是  $P(S_{n+1}^{(j)} / S_n^{(i)})$  和  $P(\mathbf{X}_n / S_n^{(i)})$ 。各个  $\mu^{(k)}$  中有若干音子是相同的，因此各  $\mu^{(k)}$  的两套参数中有若干项是相同的。用  $\lambda$  表示全部  $\mu^{(k)}$  的所有参数的总和。这样，已知  $\mathbf{X}$  时一段话语由  $\mu^{(k)}$  产生的后验概率可以表示为

$$P(\mu^{(k)} / \mathbf{X}, \lambda)$$

特地将  $\lambda$  标示出来是为了指明各  $\mu^{(k)}$  的参数取自  $\lambda$ 。

语音识别系统的运作分成训练阶段和工作阶段。在训练阶段，首先采集到足够量的已知其内容的话语段，每一段话语以某一长度的特征向量链  $\mathbf{X}_m$  的形式存储起来，该话语的 HMM 为已知，可记为  $\mu_m$ 。设共有  $M$  段话语，则构成一个训练集  $\{\mathbf{X}_m, \mu_m\}, m = 1 \sim M$ ，

$\mu_m \in \{\mu^{(1)} \dots \mu^{(K)}\}$ 。在训练时通过一套算法求  $\lambda$  使得各  $P(\mu_m/\mathbf{X}_m, \lambda)$  达到最大。在识别阶段, 则用已求得的  $\lambda$  对于各种可能的  $\mu^{(k)}$  计算  $P(\mu^{(k)}/\mathbf{X}, \lambda)$  并求得  $k^*$ 。无论对于训练还是对于识别, 直接计算后验概率  $P(\mu^{(k)}/\mathbf{X}, \lambda)$  是困难的。为此可以利用下列的贝叶斯公式:

$$P(\mu^{(k)}/\mathbf{X}, \lambda) = \frac{P(\mathbf{X}/\mu^{(k)}, \lambda) P(\mu^{(k)})}{P(\mathbf{X}/\lambda)} \quad (2-158)$$

这样, 问题归结为求先验概率  $P(\mathbf{X}/\mu^{(k)}, \lambda)$ ,  $\mathbf{X}$  产生概率  $P(\mathbf{X}/\lambda)$  和  $\mu^{(k)}$  产生概率  $P(\mu^{(k)})$ 。最后一项取决于该段话语中各个词的词序搭配关系出现的概率, 它取决于文法, 而与各个音子的 HMM 参数无关。由于  $\lambda$  只表示音子和词的 HMM 参数, 所以该项概率不是  $\lambda$  的函数, 其计算可以通过语言学的途径进行统计分析而得到, 此处不赘。下面分别针对训练和识别讨论另外两项概率的计算。

在训练阶段需要一套训练算法来求得一套最佳参数  $\lambda^*$ , 使得下列各后验概率达到最大, 即

$$P(\mu_m/\mathbf{X}_m, \lambda^*) = \max\{P(\mu_m/\mathbf{X}_m, \lambda)\}, \quad m = 1 \sim M \quad (2-159)$$

首先, 将式 (2-158) 写成下列形式:

$$P(\mu_m/\mathbf{X}_m, \lambda) = \frac{P(\mathbf{X}_m/\mu_m, \lambda) P(\mu_m)}{P(\mathbf{X}_m/\mu_m, \lambda) + \sum_{\substack{r=1 \\ r \neq m}}^M P(\mathbf{X}_m/\mu_r, \lambda)} \quad (2-160)$$

在改写式 (2-158) 时, 令其中  $\mu^{(k)} = \mu_m, \mathbf{X} = \mathbf{X}_m$ 。为了求最佳  $\lambda$ , 即使式 (2-160) 左侧达到最大, 可以采用两种求解准则。第一种, 令此式右侧中的先验概率  $P(\mathbf{X}_m/\mu_m, \lambda)$  达到最大。这称为最大似然估计准则 (maximum likelihood estimation, MLE)。第二种, 在令  $P(\mathbf{X}_m/\mu_m, \lambda)$  达到最大的同时, 令  $\sum_{r \neq m} P(\mathbf{X}_m/\mu_r, \lambda)$  达到最小。这称为最大互信息准则 (maximum mutual information, MMI)。从改进识别效果看, MMI 明显优于 MLE, 但是从计算复杂程度看, MMI 又明显高于 MLE。过去, 在计算资源不够充分的情况下, 各种采用 HMM 框架的语音识别系统都采用 MLE。对于 MLE 求最佳  $\lambda$  的问题归结为使各项  $P(\mathbf{X}_m/\mu_m, \lambda)$  最大的计算方法。应指出, 采用人工神经网络时, 可以部分吸收 MMI 的优点, 从而使其优于一般采用 MLE 的系统, 这将在 2.16.2 小节中讨论。

在识别阶段, 最佳参数  $\lambda^*$  已经求得, 这时只要求得一个最佳的  $k^*$ , 使先验概率  $P(\mathbf{X}/\mu^{(k)}, \lambda^*)$  与  $P(\mu^{(k)})$  乘积达到最大, 就能使后验概率  $P(\mu^{(k)}/\mathbf{X}, \lambda)$  达到最大。这样, 即可直接用按 MLE 准则求得的  $\mu^{(k^*)}$  作为对于  $\mathbf{X}$  的识别结果, 其形式化表示为

$$k^* = \arg\max_k P(\mu^{(k)}/\mathbf{X}, \lambda) = \arg\max_k P(\mathbf{X}/\mu^{(k)}, \lambda^*) \cdot P(\mu^{(k)})$$

现在讨论训练和识别都涉及的先验概率  $P(\mathbf{X}/\mu^{(k)}, \lambda)$  的计算。任一模型  $\mu^{(k)}$  产生某一特征向量序列  $\mathbf{X} = \{\mathbf{X}_1, \dots, \mathbf{X}_N\}$  时系统所经历的状态序列  $\mathbf{S} = \{S_1^{(i)}, \dots, S_N^{(i)}\}$  可能是各式各样的, 设可能经历的状态序列有  $Q$  种, 可以记之为  $\mathbf{S}_q, q = 1 \sim Q$ 。那么所需的先验概率计算可以用下式进行:

$$P(\mathbf{X}/\mu^{(k)}, \lambda) = \sum_{q=1}^Q P(\mathbf{X}, \mathbf{S}_q/\mu^{(k)}, \lambda) \quad (2-161)$$

在标准的 HMM 语音识别系统中, 即按此式对系统参数进行训练并求出  $\lambda^*$  为节约计算量, 通常采用著名的 Baum-Welch 算法<sup>[32]</sup> 来完成此式的计算。在识别时, 为减小计算量,

通常用下列近似式替代式 (2-161) 来计算该先验概率：

$$P(\mathbf{X}/\mu^{(k)}, \lambda) \approx \max_q P(\mathbf{X}, \mathbf{S}_q/\mu^{(k)}, \lambda) \quad (2-162)$$

在训练中也可以用此近似式来计算，虽然它所给出的结果是次优的，但可以节约大量的计算工作。很多实验表明，在训练的参数递推计算中，如果初值选择恰当，所得的解仍具有优良性能<sup>[103]</sup>。在下面的讨论中，无论训练还是识别，都采用式 (2-162) 的近似计算。

式 (2-162) 的右侧可以采用高效的 Viterbi 算法进行。首先，对于任何可能出现的  $\mathbf{S}_q = \{S_{q1}^{(i_1)}, \dots, S_{qN}^{(i_N)}\}$ ，下列公式成立（注意  $\mathbf{X} = \{\mathbf{X}_1 \dots \mathbf{X}_N\}$ ）：

$$\lg P(\mathbf{X}, \mathbf{S}_q/\mu^{(k)}, \lambda) = \sum_{n=1}^N \{\lg P(S_{qn}^{(i_n)}/S_{qn-1}^{(i_{n-1})}, \mu^{(k)}, \lambda) + \lg P(\mathbf{X}_n/S_{qn}^{(i_n)}, \mu^{(k)}, \lambda)\} \quad (2-163)$$

其中  $P(S_1^{(i_1)}/S_0^{(i_0)}) = P(S_1^{(i_1)})$ ，它是状态链的第一状态取某个状态的概率。这样，式 (2-162) 右侧可以按下列程序来计算。

设  $\mu^{(k)}$  中共含  $I$  个状态，记为  $\{S^{(i)}, i = 1 \sim I\}$ 。在状态链的每一时序  $n$  系统所处状态  $S_n^{(i_n)}$  可以是这  $I$  个状态中的任意一个。这样从 1 开始，到任何一个  $n$  值为止的部分状态链共有  $I^n$  种。假设对于任何  $n > 0$  已经求得了以  $S_{n-1}^{(i_{n-1})}, i_{n-1} = 1 \sim I$  为终止态的  $I$  条最佳部分状态链。所谓最佳是指这  $I$  条链的对数概率和  $\varphi_{n-1}^{(i_{n-1})}, i_{n-1} = 1 \sim I$  较之其他可能出现部分的链都要大。其计算公式是

$$\varphi_{n-1}^{(i_{n-1})} = \max_{\substack{S_1^{(i_1)}, \dots, S_{n-2}^{(i_{n-2})} \\ S_{n-1}^{(i_{n-1})} \text{ 固定}}} \left\{ \sum_{m=1}^{N-1} [\lg P(S_m^{(i_m)}/S_{m-1}^{(i_{m-1})}, \mu^{(k)}, \lambda) + \lg P(\mathbf{X}_m/S_m^{(i_m)}, \mu^{(k)}, \lambda)] \right\} \quad (2-164)$$

那么，以  $S_n^{(i_n)}, i_n = 1 \sim I$  为终止态的  $I$  条最佳部分状态链可求得如下。对任何  $i_n$ ，其前  $n-1$  个状态是以  $S_{n-1}^{(i_{n-1}^*)}$  为结尾的部分最佳状态链。 $\zeta_n(i_n)$  可以用下式求得：

$$\zeta_n(i_n) = \operatorname{argmax}_{i_{n-1}} (\varphi_{n-1}^{(i_{n-1})} + \lg P(S_n^{(i_n)}/S_{n-1}^{(i_{n-1})}, \mu^{(k)}, \lambda)) \quad (2-165)$$

而且对任何  $i_n$  可求得：

$$\varphi_n^{(i_n)} = \max_{i_{n-1}} \{\varphi_{n-1}^{(i_{n-1})} + \lg P(S_n^{(i_n)}/S_{n-1}^{(i_{n-1})}, \mu^{(k)}, \lambda)\} + \lg P(\mathbf{X}_n/S_n^{(i_n)}, \mu^{(k)}, \lambda) \quad (2-166)$$

当  $n = 1$  时， $\varphi_1^{(i_1)} = \lg P(S_1^{(i_1)}/\mu^{(k)}, \lambda) + \lg P(\mathbf{X}_1/S_1^{(i_1)}, \mu^{(k)}, \lambda), i_1 = 1 \sim I$ 。利用式 (2-166) 可以依次求得  $\varphi_2^{(i_2)}, \varphi_3^{(i_3)}, \dots, \varphi_N^{(i_N)}$ 。设  $i_N^*$  为诸  $\varphi_N^{(i_N)}$  中最大者的状态标号，即

$$i_N^* = \operatorname{argmax}_{i_N} \varphi_N^{(i_N)} \quad (2-167)$$

这样，使式 (2-162) 右侧先验概率达到最大的一条状态链即可求得为

$$S_1^{(i_1^*)} S_2^{(i_2^*)} \dots S_N^{(i_N^*)} \quad (2-168)$$

其中  $i_N$  已由式 (2-167) 求得， $i_{N-1}^*, \dots, i_2^*, i_1^*$  可用下式递推求得：

$$i_{n-1}^* = \zeta_n(i_n^*) \quad (2-169)$$

$\zeta_n(\cdot)$  在式 (2-165) 中已求得。

对于识别而言，求得了最佳状态链也就获得了与之相应的词序列，即得到了识别结果。

对于训练而言，则需要用反复迭代计算来求得一套最佳参数  $\lambda$ 。迭代从节拍  $k = 0$  开始，需设置一套初始参数  $\lambda(0)$ 。用  $\lambda(0)$  可以求得训练集中各段语音所最可能经历的状态链（式 (2-166)）。利用此链再统计出新的一套参数  $\lambda(1)$ 。此过程反复进行，直至收敛<sup>[32, 103]</sup>。

$\lambda$  中包含两组参数。一组是状态间的转移概率  $P(S_n^{(i_n)} / S_{n-1}^{(i_{n-1})}, \mu^{(k)}, \lambda)$  其统计计算与人工神经网络无关, 此处不作讨论。另一组是在某个状态  $S_n^{(i_n)}$  下产生某个特征向量  $\mathbf{X}_n$  的条件概率  $P(\mathbf{X}_n / S_n^{(i_n)}, \mu^{(k)}, \lambda)$ 。此概率估计的准确性对于识别效果的影响及计算难度都高于前一组参数。用 MLFN 来实现这一概率的估计可以在多方面优于经典的估计方法。

## 2.16.2 用 MLFN 实现 $P(\mathbf{X}_n / S_n^{(i_n)}, \mu^{(k)}, \lambda)$ 的估计

为达到条件概率估计的目的, 既可用 MLP 也可用 RBF。后者的优点是运算简单、速度快, 但是在某些实验中的性能略逊于 MLP<sup>[106,107]</sup>。本书只讨论 MLP 的应用。

设全部 HMM 系统总共包含  $I$  个状态  $S^{(i)}, i = 1 \sim I$ 。如果系统中共使用了 62 个音子, 那么最简单而有效的方案是每个音子只用一个状态表示。这时系统有 62 个状态, 即  $I = 62$ 。现在可以用一个 MLP, 它的输入是特征向量  $\mathbf{X}_n$ , 它的输出层包含 62 个神经元, 每个神经元与一个音子的状态对应, 可以记为  $S_n^{(i_n)}, i_n = 1 \sim 62$ 。若  $\mathbf{X}_n$  属于编号为  $i_a$  的音子, 则当网络输入  $\mathbf{X}_n$  时, 要求网络输出 (理想值) 为

$$S_n^{(i_a)} = \begin{cases} 1, & i_n = i_a \\ 0, & i_n \neq i_a \end{cases}$$

也就是说, 要求此网络成为一个分类器。在 2.12 节中业已证明, 当 MLP 的规模足够大, 学习中能使经验风险函数  $R_{\text{emp}}(\xi)$  达到全局极小点且训练集的规模足够大时, 按识别一分类要求训练的 MLP 可以作为一个后验概率估计器, 即当网络输入为  $\mathbf{X}_n$  时, 网络  $i_n$  端输出即等于  $P(S_n^{(i_n)} / \mathbf{X}_n), i_n = 1 \sim 62$ 。为简化起见, 条件概率中的条件  $\mu^{(k)}, \lambda$  略而不书, 下皆同此。

为了求先验概率  $P(\mathbf{X}_n / S_n^{(i_n)})$ , 可以用如下贝叶斯公式:

$$P(\mathbf{X}_n / S_n^{(i_n)}) = \frac{P(S_n^{(i_n)} / \mathbf{X}_n) P(\mathbf{X}_n)}{P(S_n^{(i_n)})}$$

其中  $P(S_n^{(i_n)})$  可以通过对训练集中已标明的各状态的出现频率进行统计而获得。应该看到, 对于不同的  $S_n^{(i_n)}$ , 它们的  $P(\mathbf{X}_n / S_n^{(i_n)})$  相对比例关系不受  $P(\mathbf{X}_n)$  影响, 因此在用 Viterbi 算法搜索最优状态链时, 无需计入该项。

在 HMM/MLP 语音识别系统训练中, 首先需要对训练集中的全部特征向量  $\mathbf{X}_n$  赋予状态标号  $i_n$  (例如, 由音韵专家用手工方法对不同的训练语音段加音子标记), 从中可以统计出状态转移概率并且对一个 MLP 进行训练。然后用训练好的 MLP 和状态转移概率对训练集进行识别, 即采用 Viterbi 算法搜索训练集中各语音段的最可能状态序列。由此可以统计出新的状态转移概率并对 MLP 进行新训练。这一过程反复交替进行, 直至收敛为止。在识别中, 则可以直接用训练好的 MLP 和最终的状态转移概率对未知的话语按 Viterbi 算法进行识别。

在 HMM 框架中用 MLP 代替传统的统计方法估计先验概率  $P(\mathbf{X}_n / S_n^{(i_n)})$  有如下优点。

(1) 如 2.12 节所述, 传统方法是基于某种假设模型的参数估计方法, 其中既存在模型不准确问题又存在病态问题。而 MLP 可以较好地解决这些问题。

(2) 在传统方法中只在一帧特征向量  $\mathbf{X}_n$  和一个状态  $S_n^{(i_n)}$  之间建立一一对应关系。实际情况是任何  $\mathbf{X}_n$  与其前后若干帧的特征向量  $\dots \mathbf{X}_{n-1}, \mathbf{X}_{n+1}, \dots$  之间存在相关性。MLP 的灵活性



使得它能在  $S_n^{(i)}$  与多个特征向量  $X_{n-c}, \dots, X_n, \dots, X_{n+c}$  之间建立对应关系 (见 2.16.3 小节)。

(3) 传统方法中, 将属于每个状态的特征向量分成互不相关的集合, 各自分别进行先验概率估计。而对 MLP 训练时, 是将上述各个集合的样本对同一个网络进行训练。由 MLP 对各个集团的样本进行同时的后验性估计, 因而更符合于前述的 MMI 准则, 而不是 MLE 准则。这使得各个状态的区分更明显且更节约完成估计所需的参数个数。

以上的这些论点已为大量实验结果所证实<sup>[109, 110, 111]</sup>。

### 2.16.3 HMM/MLP 语音识别系统举例<sup>[103, 109, 112]</sup>

此实际系统按连续语音识别方式工作, 使用两个词汇表, 各包含 1000 个词和 5000 个词。系统使用 TIMIT 数据库提供的 61 个音子作为基本识别单位, 每个音子只含 1 个状态, 每个词由若干音子串接而成 (见图 2-43 的示例)。

系统中所用的 MLP 如图 2-45 所示。它的输出层含 61 个神经元, 每个神经元与一个状态 (音子) 相应, 记为  $S^{(i)}, i = 1 \sim 61$ 。网络只含一个隐层, 在实验系统中隐层的神经元数在 500 ~ 4000 的范围内进行选择。每个语音帧的特征向量  $X_n$  有 26 个分量, 其中包括 12 个 PLP (感觉加权线性预测) 系数<sup>[109]</sup>, 12 个 PLP 系数的帧间差分, 1 个对数帧能量值, 1 个对数帧能量的帧间差分。送入 MLP 的信号由 9 个特征向量组成:  $X_{n-4}, \dots, X_n, \dots, X_{n+4}$  即输入信号共含 234 个分量。

网络按随机梯度 BP 算法进行训练, 即每输入一个训练样本就对网络的权作一次调整。由于随机梯度算法具有随机性, 因此更容易摆脱局部极小点的约束。此外, 为了改进网络的性能, 还采取了下列的一些措施。

(1) 在 HMM/MLP 大词汇表、连续语音识别系统中所用的 MLP 具有很大的规模, 其中可调自由参数的个数达到几十万甚至上百万个 (在本节所举的例子中, 如隐层含 2000 个神经元, 则网络含 590000 个自由参数)。对此网络如训练过度易出现过适应问题, 所以应采用交叉有效法进行训练 (见 2.11.1 小节)。在实施交叉有效法训练时, 如果在训练进展到某一轮 (训练集中所有样本皆使用过一次称为一轮), 该轮的测试集平均误差较前一轮增加, 则应废除本轮的权调整。然后, 将调整步幅减小 (一般减小 50%) 再进行训练。采用降步幅的交叉有效训练可以有效地提高系统的推广性能<sup>[112]</sup>。实验证明, 一般在每个步幅上只需作一轮权调整, 然后即应降幅作下一轮调整。

(2) MLP 的隐层神经元皆取 Sigmoid 函数, 而输出层神经元可取线性函数或 Sigmoid 函数。当取后者时, 用相对熵经验风险函数 (式 2-117) 替代均方函数 (式 2-116), 可以明显提高训练的收敛速度。因为当神经元取 Sigmoid 函数时, 当输出接近 0 或 1 的情况下, 后者给出的调整量将很小, 而前者 (相对熵函数) 仍具有足够大的调整量。

(3) 输出层各神经元的阈值参数的训练初值应该设置为一个较大的负数。对各种已

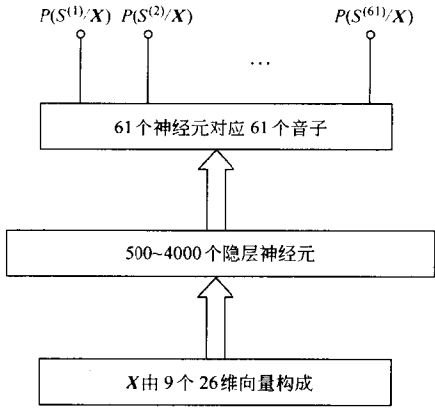


图 2-45 2.16.3 节例子所用的 MLP 示意图

训练完毕的 MLP 的输出层各神经元的阈值参数作统计分析时发现，其直方图分布在一个较负的数 其典型值为  $-4$  附近有一个强峰起。这不是偶然的。因为，在语音识别中使用的 MLP 的输出层大多取 Sigmoid 函数形式，对于某个第  $i$  输出端 当网络输入为  $\mathbf{X}_n$  该端输出应等于  $P(S_n^{(i)}/\mathbf{X}_n)$ ，若该神经元的净输入为  $I_n^{(i)}$  则  $P(S_n^{(i)}/\mathbf{X}_n) = (1 + e^{-I_n^{(i)}})^{-1}$ 。易于解出  $I_n^{(i)} = \ln[P(S_n^{(i)}/\mathbf{X}_n)/(1 - P(S_n^{(i)}/\mathbf{X}_n))]$ 。对于任何  $i$ ，就所有  $\mathbf{X}_n$  平均而言， $P(S_n^{(i)}/\mathbf{X}_n)$  总是一个远小于 1 的正数。所以  $I_n^{(i)}$  的平均值，也就是输出层第  $i$  神经元的阈值参数是一个较负的数就不足为怪了。它大致等于  $\ln[P(S_n^{(i)}/\mathbf{X}_n)]$  的平均值。如果将各阈值参数的训练初值设置在  $-4$  左右，不但可以加快训练速度而且网络的性能也略有改善。

(4) 实验结果表明，用随机梯度 BP 算法训练网络时，如果按各语音帧在话语中的原有次序送入特征向量，收敛速度较慢。若打乱原有次序随机输入，那么收敛速度较快且网络性能较佳。

(5) 在训练开始时，可以采用手工标记的各语音帧所对应的音子（状态）对网络进行训练。TIMIT 数据库的语音样本就已经用 61 个音子加了标记。

## 2.16.4 HMM/MLP 语音识别系统的实验结果

已建成了若干个 HMM/MLP 语音识别系统<sup>[103,110,113,114,115]</sup>。所有实验结果都表明了人工神经网络确能改进系统性能。下面列举一些实验的结果及其分析。

(1) 应采用规模很大的 MLP 和规模很大的训练集。MLP 的权参数在不少情况下超过了一百万个，训练集的语音帧数在几百万以上。文献<sup>[114]</sup>所给的实验系统中，MLP 用了 160 万个权，训练集含 600 万帧语音。它的词汇表容量为 5000 个词，采用标准双词文法。系统给出的测试集词误识率为 16%。若其他条件相同而 MLP 只用 40 万个权时，测试集误识率为 20%。

(2) 在可比条件大致相同时（例如，识别所用的特征向量相同，系统中可调参数的个数相同 等等）采用 MLP 的 HMM 系统较之其他 HMM 系统均表现出明显的优越性。在一个连接数字识别的实验中<sup>[111]</sup> 曾用 TI 公司提供的检验各种方案。参与比较的标准 HMM 系统采用捆绑式高斯分布模型（参考<sup>[32]</sup>）共用了 28000 个参数，数字串的误识率为 3.8%。参与比较的 HMM/MLP 系统只用了 11000 个参数，其数字串的误识率为 2.5%。当前者的参数个数增加到比后者高一个量级时，二者的误识率才比较接近。在另一个词汇表容量为 1000 的用于资源管理的连续语音识别实验中<sup>[110]</sup> 当标准 HMM 系统和 HMM/MLP 系统的参数数量相同且皆按与上下文无关的方式工作时，前者的词误识率为 11% 后者为 5.8%。如果在识别系统中引入人文约束，则二者的误识率皆可降低。但是，为了使二者的词误识率降至接近的低值，前者的参数数量仍需较后者高一个数量级<sup>[115]</sup>。

(3) 按照当前的研究水平，当标准 HMM 系统使用非常复杂的算法且参数的数量非常多时，其性能可优于当前最好的 HMM/MLP 系统，但性能超出得十分有限。例如上面举出的连接数字识别系统<sup>[111]</sup> 中 如果 HMM 系统的参数扩充到 20 万个，则数字串误识率为 2%。较之用了 1.1 万个参数的 HMM/MLP 系统 其误识率只减小了 20%。

(4) 如果将 HMM/MLP 给出的  $P(\mathbf{X}_n/S_n^{(i)})$  与标准 HMM 给出的  $P(\mathbf{X}_n/S_n^{(i)})$  进行平滑组合(即加权取平均),则相应系统的误识率较原来各系统都低。以上述 3) 中列举的连接数字识别系统为例,当标准 HMM 系统和 HMM/MLP 系统各自独立工作时,数字串误识率分别为 2% 和 2.5%,而将二者混合时,误识率降至 1.7%。在上述的用于资源管理的连续语音识别系统中<sup>[110]</sup>,如果采用非常复杂的算法和非常多的参数时,标准 HMM 系统的词误识率可降至 3.8%。如果再与 HMM/MLP 系统混合,则可以进一步降至 3.2%。上两例中显示的现象在许多其他实验中也观察到了。

## 2.16.5 HMM/MLP 系统进一步研究的课题

### 1. 在系统框架中纳入协同发音

在连续语音中,每一个音子嵌在不同的上下文(即左右音子)之间时,发音会有变化,这就是协同发音现象<sup>[32]</sup>。如果能在系统框架中引入这一影响,将能改善识别效果。在不考虑上下文影响条件下建立的每个音子的 HMM 参数称为 CI(context independent)音子参数。上列各节讨论的 MLP 就是针对 CI 音子参数的。如纳入上文(左侧音子)对本音子的影响,可建立 LCD(left context dependent)音子参数,或称 LCD 双音子模型。如考虑下文(右侧音子)影响,则可建立 RCD 双音子模型。如上下文(左右侧音子)影响都考虑,则可建立三音子(triphone)模型。在训练集的规模不够充分大的条件下,可将 CI、LCD、RCD、triphone 这 4 种模型的  $P(\mathbf{X}_n/S_n^{(i)})$  按一定的比例组合起来(加权和)作为系统的先验概率。

前面 3 种模型的  $P(\mathbf{X}_n/S_n^{(i)})$  也可以用 MLP 来实现其估计。图 2-46 给出了用 MLP 估计这 3 种模型的  $P(S_n^{(i)}/\mathbf{X}_n)$  的一种方案。求得此 posterior 概率后,再用贝叶斯公式将其转换为

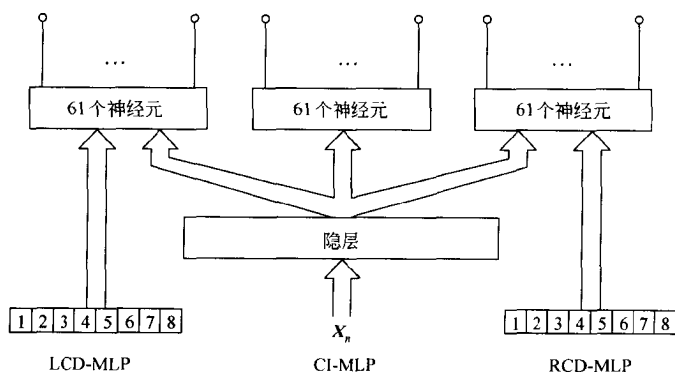


图 2-46 用 MLP 估计 3 种模型的  $P(S_n^{(i)}/\mathbf{X}_n)$  的方案

上列的先验概率。在此方案中将 LCD 的左侧音子或 RCD 的右侧因子分成 8 种类别(对于 triphone 模型也可以按照此原则处理)。此方案的 MLP 分成左中右 3 个。中间的 MLP 用于 CI 模型,其训练和运作与常规的不考虑上下文影响的 MLP 无差异。用于 LCD 模型的左侧 MLP 和用于 RCD 模型的右侧 MLP 不设置自身单独的隐层,而将中间 CI-MLP 的隐层输出作为其输出层的输入,在训练时只对这些层之间的权进行调整。此外,左右两侧各有由 8 个二进制单元构成的向量,它们也是相应输出层的输入信号。这 8 个二进制单元中每

一个与一个音子大类别对应；当左或右侧音子属于某种大类别时，相应的二进制单元取值为 1 而其他单元取值为 0。另一种方案是在中间网络的左右侧分别设置 8 个分立的输出层，当左右音子属某一大类时，即取用与之相应的输出层。这一方案的性能略优于前一方案，但是计算及存储量大。

## 2. 采用多网络法

第一种方案是用 61 个网络 每个网络与一个音子( 状态 )对应。在训练时 当  $\mathbf{X}_n$  属于某一音子，则相应网络的输出（只有一个输出端）理想值为 1 其他网络的输出理想值为 0。训练完毕后，每个网络给出各自的  $P(S^{(i)}/\mathbf{X}_n)$ 。第二种方案是设置多个 MLP，每个 MLP 仍具有 61 个输出端，但是不同的 MLP 用不同的训练集进行训练。在用于识别时，将各网络给出的输出取平均来得到所需的概率估计<sup>[117]</sup>。第三种方案是男、女说话人分别用两个不同的 MLP 进行训练和识别。

关于 MLFN 和 RNN 的研究尚有很大的潜在发展空间。以下几方面值得注意：（1）由强训练多网络或弱训练多网络组合成的 ANN 一直受到高度重视，但迄今为止还没有关于这方面研究彻底的理论阐释文献发表。文献<sup>[167]</sup>给出了一些最新研究成果。（2）新的学习算法仍有待开发。最近，借鉴自统计学领域的 EM(expectation-maximization)参数学习算法已引起很多 ANN 研究者的重视<sup>[167]</sup> 可参阅 7.11.2 小节。（3）统计学习理论仍是一个受到广泛关注的领域，其中 VCdim 的估计，支持向量机(SVM)的构成和应用等问题都是研究重点<sup>[168]</sup>。（4）如何将进化算法用于 MLFN 和 RNN 的学习仍有大量工作可做，这将在第 6 章中介绍。

## 参考文献

- [1] 杨行峻, 郑君里. 人工神经网络. 北京: 高等教育出版社, 1992
- [2] Cybenko G. Approximation by superpositions of sigmoidal function. *Mathematics of Control, Signals and Systems*, 1989, 2(4): 303~314
- [3] Schmetterer L. Multidimensional stochastic approximation. In: Krishnaiah, ed. *Multivariate Analysis II*. New York: Academic, 1969
- [4] Albert A E, Gardner L A, Jr. *Stochastic approximation and nonlinear regression*. Cambridge, MA: MIT Press, 1967
- [5] Rumelhart D E, McClelland J L. *Parallel distributed processing Vol. 1 and Vol. 2*. Cambridge, MA: MIT Press, 1986
- [6] 连小珉. 前向多层 ANN 的双重学习算法: [内部论文]. 北京: 清华大学, 1995
- [7] Baba Norial, et al. A Hybrid algorithm for finding the global minimum for error function of neural networks and its applications. *Neural Networks*, 1994, 7(8) : 1253~1265
- [8] Maniezzo Vittorio. Genetic evolution of the topology and weight distribution of neural networks. *IEEE Trans. on Neural Networks*, 1994, 5(1)
- [9] Hancock P J B. Recombination operators for the design of neural nets by genetic algorithm. In: Manner R et al, ed. *Parallel Problem Solving from Nature, 2*. Amsterdam: Elsevier, 1992
- [10] Kilano H. Empirical studies on the speed of convergence of neural network training using genetic algorithms. In: *Proc. of Eighth Nat. Conf. of AI(AAAI-90)*. 1990. 789~799
- [11] Maniczzo V. Searching among search space: Hastening the genetic evolution of feed forward neural networks. In: *Int. Conf. on Neural Networks and Genetic Algorithms, GA-ANN'93*. Heidelberg: Springer-Verlag, 1993. 635~642
- [12] Yao Xin. A Review of evolutionary artificial neural networks. *International Journal of Intelligence Systems*, 1993, 8: 539~567
- [13] Yoon Byungjoo, et al. Efficient genetic algorithm for training layered feedforward neural networks. *Information Sciences*, 1994, 76: 67~85
- [14] Saha Swapan, et al. Genetic design of sparse feedforward neural networks. *Information Sciences*
- [15] Harp Steven A, et al. Genetic synthesis of neural network architecture. In: L. Davis, ed. *Handbook of Genetic Algorithms*. New York: 1991
- [16] Kirkpatrick S, et al. Optimization by simulated annealing. *Science*, May 1983, 220: 671~680
- [17] Ergezinger S, et al. An accelerated learning algorithm for multilayer perceptrons, optimization layer by layer. *IEEE Trans. on Neural Networks*, Jan. 1995, 6(1)
- [18] Baldi Pierre. Gradient descent learning algorithm overview; a general dynamical systems perspective. *IEEE Trans. on Neural Networks*, Jan. 1995, 6(1)
- [19] Hush Don R, et al. Progress in supervised neural Networks—what's new since Lippmann? *IEEE Signal Processing Magazine*, Jan. 1993
- [20] Waymaere Nico, et al. On the initialization and optimization of multilayer perceptrons. *IEEE Trans. on Neural Networks*, Sep. 1994. 5(5)
- [21] Ochiai Keihero, et al. Kick-out learning algorithm to reduce the oscillation of weights. *Neural Net-*

- works, 1994, 7(5): 797~807
- [22] Sperduti Alessandro, et al. Speed up learning and network optimization with extended back propagation. *Neural Networks*, 1993, 6: 365~383
  - [23] Biegler-Köng Friedrich. A learning algorithm for multilayered neural networks based on linear least squares problems. *Neural Networks*, 1993, 6: 127~131
  - [24] Tang Zaiyong, et al. Deterministic global optimal FNN training algorithms. *Neural Networks*, 1994, 7(1): 1~11
  - [25] Perantonis S J, et al. An efficient constrained learning algorithm with momentum acceleration. *Neural Networks*, 1995, 8: 237~249
  - [26] Parlos A G, et al. An accelerated learning algorithm for multilayer perceptron networks. *IEEE Trans. on Neural Networks*, 1994, 5: 493~497
  - [27] Patrick P. Minimization methods for training feed forward neural networks. *Neural Networks*, 1994, 7: 1~11
  - [28] Shang Yi, et al. Global optimization for neural network training. *Computer*, March 1996: 45
  - [29] Villers J de, et al. Backpropagation neural nets with one and two hidden layers. *IEEE Trans. on Neural Networks*, Jan. 1992, 4: 136~141
  - [30] Obradovic Z, et al. Small depth polynomial size neural networks. *Neural Computation*, 1990, 2: 402~404
  - [31] Tamura Shin'ichi, et al. Capabilities of a four-layered feed forward neural networks; four layer versus three. *IEEE Trans. on Neural Networks*, 1997, 8: 251~255
  - [32] 杨行峻, 迟惠生等. 语音信号数字处理. 北京: 电子工业出版社, 1995
  - [33] Mulgrew Bernard. Applying radial basis functions. *IEEE Signal Processing Magazine*, 1996, 13(2): 50~65
  - [34] Broomhead D S, et al. Multivariable function interpolation and adaptive networks. *Complex Systems*, 1988, 2: 321~355
  - [35] Shepherd T J, et al. Nonlinear signal processing using radial basis functions. *SPE Advanced Signal Processing Algorithms, Architectures and Implementations*, 1990, 13: 51~61
  - [36] Chen S, et al. Nonlinear systems identification using radial basis functions. *International Journal System Science*, Dec. 1990, 21(12): 2513~2539
  - [37] Cha I, et al. Interference cancellation using radial basis function networks. In: *EURASIP Signal Processing*. Dec. 1995, 147(3): 247~268
  - [38] Chen S, et al. Reconstruction of binary signals using an adaptive radial basis function equaliser. In: *EURASIP Signal processing*. 1991, 22(1): 77~93
  - [39] Hartman E J, et al. Layered neural networks with Gaussian hidden units as universal approximation. *Neural Computation*, 1990, 2(2): 210~215
  - [40] Park I, et al. Universal approximation using radial basis function networks. *Neural Computation*, 1991, 3: 247~257
  - [41] Park I, et al. Approximation and radial basis function networks. *Neural Computation*, 1993, 5: 305~316
  - [42] Chen S, et al. Clustering technique for digital communications channel equalization using radial basis function networks. *IEEE Trans. on Neural Networks*, 1993, 4(4): 570~580
  - [43] Whitehead B A, et al. Cooperative-competitive genetic evolution of radial basis function centers and

- widths for time series prediction. *IEEE Trans. on Neural Networks*, 1996, 7(4): 869~881
- [44] Bors A G, et al. Median radial basis function neural network. *IEEE Trans. on Neural Networks*, 1996, 7(6): 1351~1365
  - [45] Elanayar S, et al. Radial basis function neural network for approximation and estimation of nonlinear stochastic dynamic systems. *IEEE Trans. on Neural Networks*, 1994, 4(5): 594~604
  - [46] Lee S, et al. Unconstrained handwritten numeral recognition based on radial basis competitive and cooperative networks with spatio-temporal feature representation. *IEEE Trans. on Neural Networks*, 1996, 7(2): 455~475
  - [47] Krzyzak A, et al. Nonparametric estimation and classification using radial basis function nets and empirical risk minimization. *IEEE Trans. on Neural Networks*, 1996, 7(2): 475~488
  - [48] Zhang J, et al. Wavelet neural networks for function learning. *IEEE Trans. on Signal Processing*, 1995, 43(6): 1485~1497
  - [49] Pati Y C, et al. Analysis and synthesis of feedforward neural networks using discrete affine wavelet transformations. *IEEE Trans. on Neural Networks*, 1993, 4(1): 73~85
  - [50] Delyon B, et al. Accuracy analysis for wavelet approximations. *IEEE Trans. on Neural Networks*, 1995, 6(2): 332~347
  - [51] Zhang Q, et al. Wavelet networks. *IEEE Trans. on Neural Networks*, 1992, 3(6): 889~898
  - [52] Meyer Y. Wavelets; algorithms and applications. Philadelphia, PA; SIAM Press, 1993
  - [53] Chui C K. An introduction to wavelets. New York; Academic Press, 1992
  - [54] Rioul Olivier, et al. Wavelets and signal processing. *IEEE SP Magazine*, Oct. 1991: 14~38
  - [55] Walter Gilbert G. Approximation of delta function by wavelets. *J. Approx. Theory*. 1992, 71: 329~343
  - [56] Scott D W. Multivariate density estimation. New York; Wiley, 1992
  - [57] Huang J N, et al. Regression modelling in back-propagation and projection pursuit, learning. *IEEE Trans. on Neural Networks*, 1994, 5(3): 342~353
  - [58] Oppenheim A V, Schaffer R W. Digital signal processing. Englewood Cliffs; Prentice-Hall, 1975
  - [59] Cohen Leon. Time-frequency distribution—a review. *Proc. IEEE*, 1989, 77(7): 941~981
  - [60] Haykin Simon. Neural networks expand SP's horizons. *IEEE Signal Processing Magazine*, 1996, 13(2): 24~49
  - [61] Lo S-C B, et al. Wavelet-based convolution neural network for pattern recognition. *World Congress on Neural Networks*, 1995, 2: 838~843
  - [62] Szu H, et al. Remote ECG diagnosis using wavelet transform and artificial neural networks. *World Congress on Neural Networks*, 1995, 2: 844~848
  - [63] Xu Lei, et al. Robust principle component analysis by self-organizing rules based on statistical physics approach. *IEEE Trans. on Neural Networks*, 1995, 6(1): 131~143
  - [64] Oja E. A simplified neuron model as a principle component analyzer. *Journal of Mathematical Biology*, 1982, 15: 267~273
  - [65] Sanger T D. Optimal unsupervised learning in a single-layer feedforward neural network. *Neural Networks*, 1989, 1: 459~473
  - [66] Barron A R. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. on Information Theory*, 1993, 39(3): 930~945
  - [67] Corradi V, et al. Regularized neural networks; some convergence rate results; [preprint]. 1993

- [68] Xu L, et al. On radial basis function nets and kernel regression; statistical consistency, convergence rates and receptive field size; [preprint]. 1993
- [69] Vapnik V N. The nature of statistical learning theory. Berlin; Springer-Verlag, 1995
- [70] Baum E B, Haussler D. What size net gives valid generalization? .Neural Computation, 1989, 1;151~160
- [71] Sontag E D. Sigmoid distinguish more effectively than heavisides. Neural Computation, 1989, 1;470~472
- [72] Tikhonov A N. On solving ill-posed problem and method of regularization. Doklady Akademii Nauk USSR, 1963, 153;501~504
- [73] Tikhonov A N, Arsenin V Y. Solution of ill-posed problem. W. H. Winston, Washington DC. ;1973
- [74] Ishikawa M. A Structural learning algorithm with forgetting of link weights. In; Tech. Rep. TR-90-7. Electrotechnical Lab. Tsukuba-City; 1990
- [75] Nowlan S T, Hinton G E. Simplifying neural networks by soft weight-sharing. Neural Computation, 1992, 4(4);473~493
- [76] Segee B E, Carter M J. Fault tolerance of pruned multilayer networks. In; Proc. Int. Joint Conf. Neural Networks; Seattle. 1991. 2;447~452
- [77] Weigend A S, et al. Generalization by weight-elimination applied to currency exchange rate prediction. In; Proc. Int. Joint Conf. Neural Networks Seattle; 1991, 1;837~841
- [78] Chauvin, Y. A back-propagation algorithm with optimal use of hidden units. In; D. S. Touretzky, ed. Advances in Neural Information Processing(1). Denver; 1989;519~526
- [79] Chauvin Y. Generalization performance of over-trained back-propagation networks. In; Neural Networks. Proc. EUROSIP Workshop, Feb. 1990. 46~55
- [80] Reed Russel. Pruning algorithms-a survey. IEEE Trans. on Neural Networks, 1993, 4(5);740~747
- [81] Mozer M C, et al. Skeletonization; a technique for trimming the fat from a network via relevance assessment. In; Touretzky D S, ed. Advances in Neural Information Processing (1). Denver; 1989. 107~115
- [82] Segee B E, et al. Fault tolerance of pruned multilayer networks. In; Proc. Int. Joint Conf. Neural Networks. 1991. 2;447~452
- [83] Karnin E D. A simple procedure for pruning back-propagation trained neural networks. IEEE Trans. on Neural Networks, 1990, 1(2);239~242
- [84] Cun Y Le, et al. Optimal brain damage. In; Touretzky D S, ed. Advances in Neural Information Processing(2). Denver; 1990. 598~605
- [85] Krushke J K. Creating local and distributed bottle necks in hidden layers of back-propagation network. In; D. Touretzky, ed. Proc. 1988 Connectionist Models Summer School. 120~126
- [86] Cun Y Le, et al. Backpropagation applied to handwritten zip code recognition. Neural Computation. 1989, 1(4);541~551
- [87] Waibel A, et al. phoneme recognition using time-delay neural networks. IEEE Trans. on ASSP. 1989, 37(3);328~339
- [88] Read R, et al. Similarities of error regularization, sigmoid gain scaling, target smoothing and training with jitter. IEEE Trans. on Neural Networks, 1995, 6(3);529~538
- [89] Sarkar D. Randomness in generalization ability; a source to improve it. IEEE Trans. on Neural Networks, 1996, 7(3);676~685



- [90] Pearlmutter B A, et al. Chaitin-Kolmogorov complexity and generalization in neural networks. In: Advances in Neural Information Processing Systems 3. San Mateo, CA; Morgan Kaufmann
- [91] Drucker H, et al. Improving performance in neural networks using a boosting algorithm. In: Advances in Neural Information Processing Systems 5. San Mateo, CA; Morgan Kaufmann
- [92] Perrone M P, et al. When networks disagree; ensemble methods for hybrid networks. In: Mammon R J, ed. Neural Networks for Speech and Image Processing. London; Chapman-Hall, 1993
- [93] Hanse L K, et al. Neural networks ensembles. IEEE Trans. on PAMI, 1990, 12; 993~1001
- [94] Jakobs R A, et al. Adaptive mixtures of local experts. Neural Computation, 1991, 3; 79~87
- [95] Wolpert D. Stacked generalization. Neural Networks, 1992, 5; 241~259
- [96] Cun Le, et al. Handwritten digit recognition with back propagation networks. In: Advances in Neural Information Processing Systems 2. Morgan Kaufmann, 1990. 396~404
- [97] Cun Le, et al. Handwritten zipcode recognition with multilayer networks. In: Proc. of Inter. Conf. on Pattern Recognition. Atlantic City; 1990. 34~39
- [98] Bottou L, et al. Comparison of classifier methods; a study in handwritten digit recognition. In: Proc. 12th IAPR, Int. Conf. on Pattern Recognition 2, 1994. 77~83
- [99] Bottou L. et al. Local learning algorithm. Neural Computation, 1992, 4; 888~900
- [100] Druckner H, et al. Boosting performance in neural networks. International Journal in PRAI, 1993, 7(4); 705~719
- [101] Simard P, Y et al. Efficient pattern recognition using a new transformation distance. In: Neural Information Processing Systems, 5. Morgan Kaufmann. 1993, 50~58
- [102] Waibel A, et al. Phoneme recognition; neural networks vs. hidden Markov models. In: Proc. IEEE Int. Conf. on ASSP. 1988; 107~110
- [103] Morgan N, Bourlard H. Continuous speech recognition. IEEE Signal Processing Magazine, 1995 12(3); 25~42
- [104] Bourlard H, Morgan N. Continuous speech recognition by connectionist statistical methods. IEEE Trans. on Neural Networks, 1993, 4(6); 893~910
- [105] Lamel L F, et al. Speech data base development; design and analysis of the acoustic-phonetic corpus. In: Proc. of the DARPA Speech Recognition Workshop, 1986. 100~109
- [106] Renals S, et al. Connectionist optimization of tied mixture hidden Markov models. In: Mooney J et al, ed. Advances in Neural Information Processing Systems, 4, 1992. 167~174
- [107] Singer E, Lippmann R A. speech recognizer using radial basis function neural networks in a HMM framework. In: Proc. IEEE Int. Conf. on ASSP. 1992. 629~632
- [108] Richard M D, Lippmann R P. Neural network Classifiers estimate Bayesian a posteriori probabilities. Neural Computation, 1991, 3; 461~483
- [109] Morgan M, et al. Continuous speech recognition using PLP analysis with multilayer perceptron. In: Proc. IEEE Int. Conf. on ASSP. 1991. 49~52
- [110] Renals, et al. Connectionist probability estimators in HMM speech recognizer, IEEE Trans. on Speech and Audio Processing. 1994, 2(1); 161~174
- [111] Lubensky D M, et al. Connected digit recognition using connectionist probability estimators and mixture-gaussian densities. In: IEEE Proc. Int. Conf. on Spoken Language Processing. 1994. 295~298
- [112] Morgan N, Bourlard H. Generalization and parameter estimation in feedforward nets; some experi-

- ments. In: Touretzky D S, ed. *Advances in Neural Information Processing Systems 2*. 1990
- [113] Zavaliagkos G, et al. A hybrid segmental neural net /hidden Markov model system for continuous speech recognition. *IEEE Trans. on Speech and Audio Processing*, 1994, 2(1):151~160
- [114] Morgan N. Big dumb neural nets(BDNN): a working brute force approach to speech recognition. In: *Proc. of the ICNN. 1994. VII*:4462~4465
- [115] Robinson T, et al. A neural network based, speaker independent, large vocabulary, continuous speech recognition system: The WERNICKE Project. In: *Proc. EUROSPEECH'93*. Berlin; 1993. 1941~1945
- [116] Cohen M, et al. Context-dependent multiple distribution phonetic modeling. In: *Advances in Neural Information Processing Systems*. 1993. 5:649~657
- [117] Mirghafori, et al. Parallel training of MLP probability estimators for speech recognition: a gender-based approach. In: *IEEE Workshop on Neural Networks for Signal Processing*. Greece; 1994. 289~298
- [118] Levin A U, Narendra K S. Control of nonlinear dynamical systems using neural networks; controllability and stabilization. *IEEE Trans. on Neural Networks*, 1993, 4(2):192~206
- [119] Levin A U, Narendra K S. Control of nonlinear systems using neural networks-part II: observability, identification and control. *IEEE Trans. on Neural Networks*, 1996, 7(1):30~42
- [120] Sontag E D. *Mathematical control theory*. New York: Springer-Verlag, 1990
- [121] Gorr W L. Research prospective on neural network forecasting. *International Journal of Forecasting*, 1994, 10:1~4
- [122] Hill T, et al. Artificial neural network models for forecasting and decision. *International Journal of Forecasting*, 1994, 10:5~15
- [123] Gooijer De, Kummar K. Some recent developments in non-linear modeling, testing and forecasting. *International Journal of Forecasting*, 1992, 8:135~156
- [124] Chatfield C. Neural networks: forecasting breakthrough or passing fad. *International Journal of Forecasting*, 1993, 9:1~3
- [125] Refenes A N. Comments on neural networks: forecasting breakthrough or passing fad. *International Journal of Forecasting*, 1994, 10:43~46
- [126] Callen J A. Neural network forecasting of quarterly accounting earnings. *International Journal of Forecasting*, 1996, 12(4)
- [127] Donaldson R G, et al. Forecasting Combining with neural networks. *International Journal of Forecasting*, 1996, 12(1):49~61
- [128] Dawls R, et al. The past and the future of forecasting research. *International Journal of Forecasting*, 1994, 10:151~159
- [129] Kuan C, White H. Artificial neural networks; an econometric perspective. *Econometric Review*, 1994, 13:1~91
- [130] Kwon T M, Feroz E H. A multilayer perceptron approach to prediction of the SEC's investigation target. *IEEE Trans. on Neural Networks*, 1996, 7(5):1286~1290
- [131] Cottrell M, et al. Neural modelling time series: a statistical stepwise method for weight elimination. *IEEE Trans. on Neural Networks*, 1995, 6(6):1355~1364
- [132] Connor J J. A robust neural network filter for electricity demand prediction. *J. Forecasting*, 1996, 12(6)

- [133] 吴新根, 葛家理应用人工神经网络预测油田产量. 石油勘探与开发, 1994,21(3)
- [134] Faller W E, Scherech S J. Real-time prediction of unsteady aerodynamics; application for aircraft control and maneuverability enhancement. IEEE Trans. on Neural Networks, 1995, 6(6):1461~1468
- [135] Park Y R, et al. Prediction sun spots using layered perceptron neural network. IEEE Trans. on Neural Networks, 1996,7(2):501~505
- [136] Whitehead B A, choate T D. Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction. IEEE Trans. on Neural Networks, 1996,7(4):869~880
- [137] Behera L, et al. On adaptive trajectory tracking of robot manipulator using inversion of it's neural emulator. IEEE Trans. on Neural Networks, 1996,7(6):1401~1414
- [138] Elanayar S, shin Y C. Radial basis function neural network for approximation and estimation of nonlinear stochastic dynamic systems. IEEE Trans. on Neural Networks, 1994,5(4):594~603
- [139] Lewis F L, et al. Multilayer neural-net robot controller with guaranteed tracking performance. IEEE Trans. on Neural Networks, 1996,7(2):388~399
- [140] Parlos A G, et al. Application of recurrent multilayer perceptron in modelling complex process dynamics. IEEE Trans. on Neural Networks, 1994,5(2):255~266
- [141] Connor J T, et al. Recurrent neural networks and robust time series prediction. IEEE Trans. on Neural Networks, 1994,5(2):240~254
- [142] Kechriotis G, et al. Using recurrent neural networks for adaptive communication channel equalization. IEEE Trans. on Neural Networks, 1994,5(2):267~278
- [143] Puskorius G V, et al. Neural control of nonlinear dynamical systems with Kalman filter-trained recurrent networks. IEEE Trans. on Neural Networks, 1994,5(2):279~297
- [144] Robinson T. An application of recurrent nets to phone probability estimation. IEEE Trans. on Neural Networks, 1994,5(2):298~305
- [145] Bengis Y, et al. Learning long-term dependencies with gradient descent is difficult. IEEE Trans. on Neural Networks, 1994,5(2):157~166
- [146] Nerrand O, et al. Training recurrent neural networks; why and how? an illuztration in dynamical process modelling. IEEE Trans. on Neural Networks, 1994,5(2):178~184
- [147] Morris R J T, et al. Neural Network control of communication systems. IEEE Trans. on Neural Networks, 1994,5(4):639~650
- [148] Frosini A, et al. A neural network-based model for paper currency recognition and varification. IEEE Trans. on Neural Networks, 1996,7(6):1482~1490
- [149] Watkins S S, et al. Executing of a remote sensing application on a custom neurocomputer. IEEE Trans. on Neural Networks, 1995,5(1):64~72
- [150] Nekovei R, Sun Y. Back-propagation network and its configuration for blood vessel detection in angisgrams. IEEE Trans. on Neural Networks, 1994,5(1):64~72
- [151] Michalopoulou Z H, et al. Performance evaluation of multilayer perceptrons in signal detection and classification. IEEE Trans. on Neural Networks, 1995,6(2):381~386
- [152] Rosenblum M, Davis L S. An improved radial basis function network for visual autonomous road following. IEEE Trans. on Neural Networks, 1996,7(5):1111~1120
- [153] Resenblum M, et al. Human expression recognition from motion using a radial basis function network architecture. IEEE Trans. on Neural Networks, 1996,7(5):1121~1138

- [154] Phoha V V, Oldham W J B. Image recovery and segmentation using competitive learning in a layered network. *IEEE Trans. on Neural Networks*, 1996, 7(4):843~856
- [155] 国九英. 利用误差反向传播神经网络进行油气横向预测: [内部报告]. 北京: 清华大学, 1997
- [156] 袁东风. BP 模型及其在纠错编码技术中的应用: [内部报告]. 北京: 清华大学, 1996
- [157] Werbos Paul J. Backpropagation through time; what it does and how to do it. *Proceedings of the IEEE*, Oct. 1990, 78(10): 1550~1560
- [158] Singhal S, Wu L. Training multilayer perceptrons with the extended Kalman algorithm. In: Touretzky D S, ed. *Advances in Neural Information Processing Systems 1*. Morgan Kaufmann, 1989. 133~140
- [159] Mathews M B. Neural network nonlinear adaptive filtering using the extended Kalman filter algorithm. In: *Proceedings of the International Neural Networks Conf. Paris*, 1990. I: 115~119
- [160] Williams R J. Training recurrent networks using the extended kalman filter. In: *International Joint Conf. on Neural Networks*. Baltimore, 1992, IV: 241~246
- [161] Puskorius G V, et al. Dynamic neural network methods applied to on-vehicle idle speed control. *Proc. IEEE*, 1996, 84(10): 1407~1420
- [162] Puskorius G V, Feldkamp L A. Decoupled extended Kalman filter training of feedforward layered networks. In: *International Joint Conference on Neural Networks*. Seattle, 1991. I: 771~777
- [163] Jordan M I. Generic constraints on underspecified target trajectories. In: *International Joint Conference on Neural Networks*. Washington D. C., 1989. I: 217~225
- [164] Saad Emad W, et al. Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks. *IEEE Trans. on Neural Networks*, 1998, 9(6): 1456~1469
- [165] Donaldson R G, Kamstra M. Forecasting combining with neural networks. *J. Forecasting*, 1996, 15(1): 49~61
- [166] Ruck Dennis W, et al. Comparative analysis of backpropagation and the extended Kalman filter for training multilayer perceptrons. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, June 1992, 14(6): 686~691
- [167] Ma S, Ji C. Performance and efficiency; recent advances in supervised learning. *Proceedings of the IEEE*, Sep. 1999, 87(9): 1519~1535
- [168] Vapnik Vladimir N. An overview of statistical learning theory. *IEEE Trans. on Neural Networks*, Sep. 1999, 10(5): 988~999

# 第 3 章 自组织神经网络 —— SOM 和 ART

## 3.1 概 述

自组织 (self-organized) 学习算法是一种无监督 (unsupervised) 学习算法, 在学习时, 训练集只包含输入向量训练样本而不含相应的理想输出训练样本, 这与有监督学习算法二者皆包含的情况相互对照。自组织神经网络是一种基于自组织学习算法的人工神经网络, 它与基于有监督学习算法的 MLFN 等人工神经网络一起构成这一学科的主要研究对象。

自组织神经网络的功能是将输入向量的赋值空间划分成若干子空间, 每个子空间对应于网络若干个输出端中的某一个。当输入向量属于某个子空间时, 相应输出端的取值为 1 而其他输出端的取值为 0, 这样, 根据网络的输出, 便能立即判断输入向量属于哪一个子空间。如果对各输出端进行编号, 则每个子空间具有相应的编号, 这称为其标号。子空间的划分以及子空间与各输出端的对应关系不是预先规定而是根据某种准则通过学习加以确定, 这就是自组织学习的原理, 它完全区别于由“教师”给出理想输出样本的有监督学习。

对输入空间进行划分并赋予标号的作用可以从不同角度来解释。第一, 将输入空间中具有稠密分布的向量用有限多个离散标号来表示, 实现了数据压缩 (或称为编码)。这和熟知的向量量化 (VQ) 有许多共同之处, 也有不少差异。第二, 将具有连续或稠密分布的输入向量用数量有限的标号来表示类似于人类认知领域中的概念形成或类别区分, 每个标号表示一群具有相似特征的输入向量的类别或概念的“名称”。第三, 对输入空间的划分可以直接或间接用于各种模式识别任务。以上三方面将在下文中结合具体网络予以说明。

现在研究得比较充分且得到广泛应用的两种自组织神经网络是 SOM 和 ART。SOM 是 self-organized feature mapping (自组织特征映射) 的缩写, 由芬兰科学家 T. Kohonen 提出<sup>[1,2]</sup>。这种网络的出发点是模仿动物和人的大脑皮层中具有自组织特征的神经信号传送过程<sup>[3]</sup>。随着研究工作的进展, 原来十分复杂的仿生学习算法逐渐变得简明精练, 同时对于学习算法的各种策略和参数选择也有更加透彻的了解。SOM 作为一种自组织神经网络, 其主要功能是实现数据压缩、编码和聚类, 这种类型的功能往往可以与人脑的联想记忆功能联系起来。在自组织学习的基础上, 还可以对 SOM 施行有监督学习, 以实现识别、分类等功能。相应的算法称为 LVQ (learning vector quantization)。以后又发展出了具有更强功能的各类 LVQ2 算法。SOM 的实际应用包括: 过程和系统分析 (如系统辨识、故障诊断等), 统计模式识别 (如计算机视觉、纹理分析和分类、语音识别等), DM, 机器人, 通信 (如自适应信号检测和信道均衡、图像编码等)<sup>[2]</sup>。ART 是 adaptive resonance theory

的缩写，由美国科学家 G. A. Carpenter 和 S. Grossberg 提出<sup>[4,5]</sup>。他们对人类认知的特点进行了研究（诸如集中注意力、记忆的弹性和刚性，认知过程中从底向上和由顶向下的双向作用等），在此基础上设计出对于输入模式进行自组织分类且具有人类认知特点的神经网络即 ART。最初提出的 ART 包括 ART1(针对二进制输入模式)，ART2 针对模拟输入模式 和 ART3 利用神经元突触的机制)，详见文献[3] 的介绍。在进一步的研究中提出了 ART2A和模糊(Fuzzy)ART 两种网络<sup>[7~9]</sup>，由于其功能和实用性更强且算法表述更精练，所以当前无论从研究和应用看，主要是针对这两种网络。本章将介绍 ART2A、FuzzyART 的有关问题留待第 5 章讨论。ART2A 的主要用途是对输入信号向量进行模式分类、聚类 这在人工智能、系统辨识和故障诊断、数据压缩 编码 等许多领域有实用价值。

本章 3.2 节讨论 SOM 的自组织学习算法 3.3 节讨论该算法的参数优化和自适应，3.4 节讨论 LVQ 算法 3.5 节讨论 SOM 的若干应用实例 3.6 节讨论 ART 的基本原理 3.7 节讨论 ART2A 的几种不同形式及其学习算法，3.8 节介绍 ART2A 的应用实例。

### 3.2 SOM 的结构和自组织学习算法

#### 3.2.1 SOM 的结构

SOM是由若干个神经元构成的阵列，其维数可取任一值，但是研究及实用最多的是 2 维阵列，所以下面只讨论具有 2 维阵列结构的 SOM。设阵列中共含  $M$  个神经元，它们按照行和列有序地排列在一个平面中。一种排列方式是每行每列神经元都相互对齐，如图 3-1(a) 示；另一种方式是每行对齐，隔行错开半个神经元间距，如图 3-1(b) 所示。

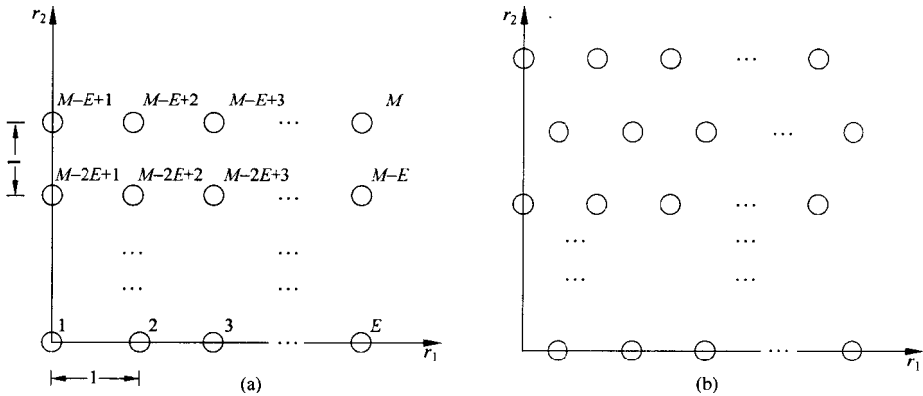


图 3-1 具有 2 维阵列结构的 SOM

由于这两种形式的应用效果差别不大，为简明计下面只讨论前一种。设阵列每行每列都含  $E$  个神经元，则神经元总数为  $M = E^2$ ，可以按行（或列）对各神经元赋予编号，图 3-1(a) 所示为按行编号的情况，此编号用  $j$  表示。设相邻神经元的行距和列距都是 1 如果设定一个 2 维坐标系，行向坐标为  $r_1$  列向坐标为  $r_2$ ，且以阵列中最左下角的神经元为坐标原点，那么每一个神经元  $j$  在阵列中的位置可以用其 2 维坐标表示为  $j(r_1, r_2)$ 。例如 编号为

$M-5E+3$  的神经元的坐标为  $r_1 = 2, r_2 = E-5$ 。

设网络的输入是  $N$  维列向量  $\mathbf{X} = [x_1, x_2, \dots, x_N]^T$  如不特别声明 其中各分量  $x_i$  取实数值。网络各神经元的输出记为  $y_j, j$  是神经元的编号,  $y_j$  只能取值为 0 或 1。每个神经元  $j$  有一个相应的  $N$  维权向量  $\mathbf{W}_j = [w_{j1}, w_{j2}, \dots, w_{jN}]^T$  各  $w_{ji}$  皆取实数值。如果各权向量已通过学习加以确定, 那么对于任一输入  $\mathbf{X}$  网络的各输出  $y_j$  可用下式计算:

$$\text{其中} \quad y_j = \begin{cases} 1, & j = j^* \\ 0, & j \neq j^* \end{cases} \quad (3-1)$$

$$j^* = \underset{j}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{W}_j\| \quad (3-2)$$

$j^*$  称为获胜神经元编号, 简称获胜者。获胜者的权向量  $\mathbf{W}_{j^*}$  与输入向量  $\mathbf{X}$  的欧氏距离小于其他非获胜者。获胜者的编号即称为该输入向量的标号。

网络的学习是通过向网络馈送训练集中的输入向量来调整各个  $\mathbf{W}_j$  使得  $\mathbf{X}$  至其标号  $j^*$  的映射满足以下两项要求。

(1) 编码误差最小

设  $\mathbf{X}$  的概率密度函数 pdf 是  $p(\mathbf{X})$  对应于  $\mathbf{X}$  的网络获胜权向量用  $\mathbf{W}_{j^*}(\mathbf{X})$  表示, 那么编码误差  $J$  可表述为

$$J = \int \|\mathbf{X} - \mathbf{W}_{j^*}(\mathbf{X})\|^2 p(\mathbf{X}) d\mathbf{X} \quad (3-3)$$

它是按照均方意义定义的。如果  $\mathbf{X}$  只能取有限个离散样本值  $\mathbf{X}_l, l = 1 \sim L, \mathbf{X}_l$  出现概率为  $p_l$  则  $J$  由下式表述:

$$J = \sum_{l=1}^L \|\mathbf{X}_l - \mathbf{W}_{j^*}(\mathbf{X}_l)\|^2 p_l \quad (3-4)$$

网络的学习应使  $J$  达到最小。如果采用有限多个样本的训练集  $\mathbf{X}_p, p = 1 \sim P (P \text{ 为样本个数})$ , 则可以用下列经验编码误差  $\hat{J}$  来替代  $J$ :

$$\hat{J} = \sum_{p=1}^P \|\mathbf{X}_p - \mathbf{W}_{j^*}(\mathbf{X}_p)\|^2 \quad (3-5)$$

当  $P$  足够大且样本集选取恰当时, 即可通过求  $\hat{J}$  最小化近似求得各最优权向量。

(2) 实现保持拓扑特性的映射

这指的是, 设有两个输入向量  $\mathbf{X}_a$  和  $\mathbf{X}_b$ , 若二者的欧氏距离较小, 则二者相应获胜神经元在阵列中的距离也小; 反之, 则较大。设  $\mathbf{X}_a$  的获胜者编号及其坐标表为  $j_a^*(r_{1a}, r_{2a})$ ,  $\mathbf{X}_b$  的表为  $j_b^*(r_{1b}, r_{2b})$ , 那么两个获胜者在阵列中的距离为

$$d_{ab} = [(r_{1a} - r_{1b})^2 + (r_{2a} - r_{2b})^2]^{\frac{1}{2}} \quad (3-6)$$

一个保持拓扑特性的映射能够保证  $d_{ab}$  正变于  $\|\mathbf{X}_a - \mathbf{X}_b\|$ 。

这一要求在一般的聚类或 VQ 算法中是不包含的。但却是一项很重要的特性, 它使得输入向量的标号之间的“距离”也携带了有关其特点的信息, 从而能更多地了解其特征并为下步处理所使用。如何用关于各  $\mathbf{W}_j$  的函数来描述一个 SOM 保持拓扑特性的良好程度将在 3.3 节中讨论。

可以附带指出, 按照目标函数  $J$  来调整各  $\mathbf{W}_j$  使其达到最小值时, 一个自然的结果是

在  $X$  的赋值域中, pdf  $p(X)$  较大的那些区域  $W_j$  的分布较密; 反之,  $p(X)$  较小的区域  $W_j$  的分布较疏。在早期的 SOM 文献中将其称为保持 pdf 特性的映射。

### 3.2.2 SOM 的自组织学习算法

本节介绍满足 3.2.1 节两项要求的启发式算法, 从理论上尚无法严格证明此算法能实现所需的要求, 但是它具有良好的实际效果, 所以得到广泛使用。此外, 现在有一些研究工作证明了在特定条件下, 此算法可收敛于满足第 1) 项要求的局部最优解<sup>[10]</sup>。在 3.2.3 节将介绍如何对此算法中的参数进行自适应调整, 使其在满足两项要求方面具有更好的效果<sup>[11]</sup>。

设有含  $P$  个样本  $X_p$  的训练集 SOM 的自组织学习算法按下列步序进行。

(1) 随机设置初始权向量  $W_j(0), j = 1 \sim L$ 。设置最大迭代计算次数  $K(K \geq P)$ 。

(2) 按照迭代节拍  $k = 1, 2, \dots, K$ , 进行下列迭代计算:

对于每一节拍  $k$ , 由训练集中依次或随机地取出一个样本输入向量并且表示为  $X(k)$ , 然后用下列迭代公式由已知的  $W_j(k-1)$  求  $W_j(k)$

$$W_j(k) = W_j(k-1) + \alpha(k)\Delta(j, j^*(k), k)[X(k) - W_j(k-1)], \quad j = 1 \sim L \quad (3-7)$$

此式中的  $\alpha(\cdot), \Delta(\cdot)$  诸函数将在下面定义。

(3) 当  $k = K$  迭代结束并输出  $W_j(K), j = 1 \sim L$  作为学习所得的诸神经元权向量。

下面首先定义算法中的一些函数, 再对于算法中的初值和参数的选择以及再学习等问题进行讨论。

#### 1. 步幅函数 $\alpha(k)$

$\alpha(k) > 0$ 。应选择  $\alpha(0)$  为较大的正数 (例如  $\alpha(0) = 0.5$ ) 当  $k$  增大时  $\alpha(k)$  逐渐减小。可供选择的  $\alpha(k)$  函数形式很多, 下面列举几种。

$$(1) \alpha(k) = \alpha(0)/k \quad (3-8)$$

$$(2) \alpha(k) = \alpha(0)\exp[-\eta(k-1)], \quad \eta > 0 \quad (3-9)$$

$$(3) \alpha(k) = [\alpha(0) - \alpha(K)](1 - k/K) + \alpha(K) \quad (3-10)$$

在式 (3-9) 中可通过改变  $\eta$  来调节  $\alpha(k)$  随  $k$  的增加而减小的速度。在式 (3-10) 中  $\alpha(K)$  是迭代结束时的步幅值 例如 当  $\alpha(0)$  选为 0.5 时  $\alpha(K)$  可以在 0.001 ~ 0.01 之间选择。

#### 2. 邻域函数 $\Delta(j, j^*(k), k)$

首先说明  $j^*(k)$  是输入向量为  $X(k)$  且 SOM 的诸权向量为  $W_j(k-1)$  时获胜者的编号 其中  $j$  是待调神经元编号。称其为邻域函数是因为其取值决定于  $j$  神经元与  $j^*(k)$  神经元在阵列中的几何距离, 与  $j^*(k)$  较邻近的神经元  $j$  的  $\Delta(\cdot)$  的取值应大于距离较远的神经元的  $\Delta(\cdot)$  取值。 $\Delta(\cdot)$  还应是迭代节拍  $k$  的函数。下面只列举多种可供选择的邻域函数中的几种。

(1) 以  $j^*(k)$  为中心划定一个邻域, 此邻域可以是方形的也可以是圆形的。图 3-2 所示为方形情况 (图中只展示了阵列的一部分), 图中所示的 4 个虚线方框表示 4 种大小不同的邻域。可以用方框每边的神经元数来衡量其大小, 在此图中 4 个邻域的“边长”分别是 1, 3, 5, 7。邻域的大小随  $k$  而变化, 故可记之为  $\Psi^*(k)$ ,  $\Psi^*(k)$  为方形时, 可以直接用“边长” $e(k)$  来表示其大小。当邻域为圆形时,  $\Psi^*(k)$  是以  $j^*(k)$  为圆心且半径为  $r(k)$  的圆,



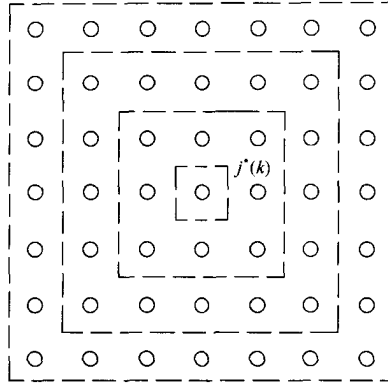


图 3-2 以  $j^*(k)$  为中心的方形邻域

这时可以用  $r(k)$  来衡量及表示其大小。无论是方形还是圆形，邻域函数均用下式计算：

$$\Lambda(j, j^*(k), k) = \begin{cases} 1, & j \in \Psi^*(k) \\ 0, & j \notin \Psi^*(k) \end{cases} \quad (3-11)$$

此式表明：只有在  $j^*(k)$  的邻域  $\Psi^*(k)$  内的那些神经元的权向量在节拍  $k$  时进行调整，邻域外的保持不变。在学习过程中，邻域随  $k$  的增大而逐渐缩小，所以一开始邻域应该包含整个阵列，最后应缩小至只包含一个神经元。邻域可按各种规律缩小，一种方案是按相等间隔阶梯形缩小。若 SOM 的正方形阵列边长为  $Q$  个神经元，当  $\Psi^*(k)$  为方形时，应选  $e(0) = 2Q - 1$  且用下式计算  $e(k)$ ：

$$e(k) = (2Q - 1) - r, \quad \frac{rK}{(2Q - 1)} \leq k < \frac{(r + 2)K}{(2Q - 1)}, \quad r = 0, 2, 4, \dots, 2Q - 2 \quad (3-12)$$

注意  $K$  是最大迭代计算次数。当  $\Psi^*(k)$  为圆形时，应选  $e(0) = \sqrt{2}(Q - 1)$ 。 $r(k)$  随  $k$  的增加除可按阶梯形缩小外，还可采用其他连续变化的形式。下面举两个例子。

$$r(k) = r(0) + [1 - r(0)] \frac{k}{K}, \quad k = 1 \sim K \quad (3-13)$$

$$r(k) = [r(0)]^{(1 - \frac{k}{K})}, \quad k = 1 \sim K \quad (3-14)$$

(2) 不划邻域，而直接用邻域函数表述邻域大小的变化。下面举两个例子。

第一种，设神经元  $j$  与获胜神经元  $j^*(k)$  在阵列中的几何（欧氏）距离为  $d_{jj^*(k)}(k)$ ，其计算公式如下：

$$d_{jj^*(k)}(k) = [(r_{1j} - r_{1j^*(k)}(k))^2 + (r_{2j} - r_{2j^*(k)}(k))^2]^{\frac{1}{2}} \quad (3-15)$$

$(r_{1j}, r_{2j})$  是  $j$  的坐标， $(r_{1j^*(k)}(k), r_{2j^*(k)}(k))$  是  $j^*(k)$  的坐标。则  $\Lambda(\cdot)$  可用下式计算：

$$\Lambda(j, j^*(k), k) = \exp \left\{ -\frac{d_{jj^*(k)}^2(k)}{2\sigma^2(k)} \right\} \quad (3-16)$$

$\sigma(k)$  称为宽度系数，其值越大相应的邻域越大。与式 (3-11) 具有明确邻域边界（边界内  $\Lambda(\cdot)$  取值为 1，边界外为 0）的情况不同，此处没有明确的边界， $\Lambda(\cdot)$  随  $d_{jj^*(k)}^2(k)$  的增大而逐渐平缓下降，其下降速度取决于  $\sigma(k)$ 。当迭代起始时应取  $\sigma(0)$  为足够大值，以使得邻域范围能覆盖整个阵列。例如阵列为边长  $Q$  神经元的矩形时，可取  $\sigma^2(0) = 2(Q - 1)^2$ 。

而当迭代趋于结束时，应取  $\sigma(K)$  为足够小值以使得邻域只覆盖  $j^*(k)$  附近。例如，可取  $\sigma^2(K) = 0.125$ 。从  $\sigma(0)$  至  $\sigma(K)$ ， $\sigma(k)$  可取阶梯、线性、指数等各种下降形式。事实上此处的  $\Delta(\cdot)$  是以  $j^*(k)$  为中心呈圆帽形逐渐下降的函数，中心点  $\Delta(\cdot)$  取值为 1 当  $d_{jj^*(k)}^2(k) = \sigma^2(k)$  时， $\Delta(\cdot)$  降至 0.6，因而  $\sigma(k)$  约相当于邻域边缘圆的半径。

第二种，仍定义  $d_{jj^*(k)}(k)$  为神经元  $j$  与  $j^*(k)$  之间的距离并用式 (3-15) 计算，则  $\Delta(\cdot)$  用下式计算：

$$\Delta(j, j^*(k), k) = \exp \left\{ -\frac{d_{jj^*(k)}(k)}{\sqrt{2}\sigma(k)} \right\}$$

$$\frac{1}{\sigma(k)} = \frac{1}{\sigma(0)} + \frac{k}{K} \left[ \frac{1}{\sigma(K)} - \frac{1}{\sigma(0)} \right] \quad (3-17)$$

对于边长为  $Q$  的方形阵列，即可令  $\sigma(0) = \sqrt{2}(Q-1)$ ， $\sigma(K) = 0.125$ 。当  $d_{jj^*(k)}(k) = \sigma(k)$  时  $\Delta(\cdot)$  值仍降至中心值的  $1/2$ ， $\sigma(k)$  相当于邻域边缘圆的半径。

第一种公式与第二种公式的区别在于：前者的  $\Delta(\cdot)$  随  $d_{jj^*(k)}(\cdot)$  的增加而按指数平方律下降，而后者按指数律下降。因此前者随  $k$  的增加而下降的速度较陡峭，而后者较平缓。此外，在式 (3-17) 中还明确给出了  $\sigma(k)$  随  $k$  而变化的规律。

### 3.2.3 SOM 自组织学习算法中的参数选择和再学习

待定的参数有阵列所含神经元数  $M$ 、最大迭代计算次数  $K$  以及各权向量的初值  $\mathbf{W}_j(0)$ 。 $M$  越大则 SOM 的编码误差  $J$  越小，但是所需的训练样本个数  $P$  也相应增加。 $P$  应较  $M$  大若干倍，例如，可在  $10M \sim 100M$  范围内选择  $P$  值。如果样本个数不受限制，则可以由  $J$  的要求确定  $M$ ，再由  $M$  确定  $P$ 。如果  $P$  的上限是确定的，则应在满足  $J$  要求的同时使  $P$  不超过其上限。 $K$  的选择与学习效果有关，经验表明， $K$  越大则学习效果越佳。但是， $K$  越大则计算开销越大。在限定计算开销的条件下，只能适当选择  $K$  值，例如可在  $30P \sim 100P$  范围内选择  $K$  值。由于编码误差  $J$  是非凸的，初值  $\mathbf{W}_j(0)$  选择不当时各  $\mathbf{W}_j(K)$  可能收敛在效果不佳的某个局部极小点上。对于普通 VQ 而言，现在有一些效果较好的初值设置策略可资参考<sup>[12]</sup>，但是如何设置使编码误差和拓扑特性保持两项要求同时得到满足的初值，尚是一个待研究的问题。

最后讨论一下再学习的问题。在前述自组织学习结束后，若再针对获胜端进行若干次学习，还可以使  $J$  值有所下降，这就是再学习。其计算公式是

$$\mathbf{W}_j(k) = \begin{cases} \mathbf{W}_j(k-1) + \alpha_{00}[\mathbf{X}(k) - \mathbf{W}_j(k-1)], & j = j^*(k) \\ \mathbf{W}_j(k-1), & j \neq j^*(k) \end{cases} \quad (3-18)$$

其中  $\alpha_{00}$  是一个较小的步幅值，例如可以令  $\alpha_{00} = 0.001$ 。

### 3.2.4 SOM 自组织学习算法收敛性的理论问题

3.2.2 节介绍的学习算法是一种启发式算法，在理论上并没有证明或回答下列一些问题。第一，当迭代次数  $K \rightarrow \infty$  时，各  $\mathbf{W}_j(k)$  是否收敛到一个良解上？即此解是否使得编码误差  $J$  达到全局最小值或足够小的某个局部最小值且保持映射的拓扑特性。第二，算法

的收敛特性与  $\alpha(k)$ ,  $\Delta(j, j^*(k), k)$  以及诸初值  $\mathbf{W}_j(0)$  的选择有何关系? 迄今为止, 对这些问题的研究大多是针对 1 维 SOM 阵列的<sup>[13~16]</sup>。而 2 维阵列的问题尚未解决, 文献[17] 采用一种修正的 SOM 学习算法<sup>[18]</sup> 来证实有关的 2 维阵列的学习收敛性。针对输入向量具有离散分布, 2 维 SOM 学习算法中的邻域  $\Psi^*(k)$  为固定圆形 (即其半径为常数) 且  $\Delta(\cdot)$  用式 (3-11) 来计算的情况, 文献[10] 证明了, 若  $\alpha(k)$  满足条件

$$\sum_{k=1}^{\infty} \alpha(k) = \infty, \quad \sum_{k=1}^{\infty} \alpha(k) < \infty \quad (3-19)$$

则诸  $\mathbf{W}_j(k)$  在  $k \rightarrow \infty$  时将收敛到稳态解  $\hat{\mathbf{W}}_j$ 。显然,  $\alpha(k) = \alpha_0/k (\alpha_0 > 0)$  满足此条件。由于采用固定圆形邻域 (设其半径为  $r_0$ ) 因此 SOM 学习算法中的编码误差  $J$  (式 (3-4)) 由下列  $J_1$  替代:

$$J_1 = \sum_{l=1}^L p_l \sum_{j=1}^M \Delta(j, j^*(\mathbf{X}_l)) \|\mathbf{X}_l - \mathbf{W}_j\|^2 \quad (3-20)$$

其中  $j^*(\mathbf{X}_l)$  是输入为  $\mathbf{X}_l$  时的获胜者编号。 $\Delta(\cdot)$  中无变量  $k$  是因为邻域函数不随  $k$  而变化。算法收敛到的稳态解是  $J_1$  的一个局部极小点, 但是在保持映射拓扑性上没有保证。当  $r_0$  给定时所收敛的稳态解  $\hat{\mathbf{W}}_j, j = 1 \sim M$ , 满足下列方程:

$$\hat{\mathbf{W}}_j = \frac{\sum_{\mathbf{X}_l \in \Omega_j} p_l \mathbf{X}_l}{\sum_{\mathbf{X}_l \in \Omega_j} p_l} \quad (3-21)$$

其中  $\Omega_j$  是由下式定义的诸  $\mathbf{X}_l$  的集合:

$$\Omega_j = \{\mathbf{X}_l : d_{jj^*}(\mathbf{X}_l) \leq r_0\}, \quad j = 1 \sim M \quad (3-22)$$

若  $r_0 \geq \sqrt{2}(Q-1)$  则所有  $\mathbf{X}_l$  都满足  $d_{jj^*}(\mathbf{X}_l) \leq r_0$  条件, 所以任何  $\Omega_j$  都包括  $\mathbf{X}_l, l = 1 \sim L$  故  $\mathbf{W}_j$  可用下式计算:

$$\hat{\mathbf{W}}_j = \frac{\sum_{l=1}^L p_l \mathbf{X}_l}{\sum_{l=1}^L p_l} = \sum_{l=1}^L p_l \mathbf{X}_l, \quad j = 1 \sim M$$

即阵列中  $M$  个权向量全相等, 它是各  $\mathbf{X}_l$  的“质心”。若  $r_0 = 0$  则  $\Omega_j$  只包含与  $\hat{\mathbf{W}}_j$  最邻近的那些  $\mathbf{X}_l$ , 即  $\Omega_j = \{\mathbf{X}_l : \|\mathbf{X}_l - \hat{\mathbf{W}}_j\| \leq \|\mathbf{X}_l - \hat{\mathbf{W}}_i\|, i \neq j\}$  这时各  $\hat{\mathbf{W}}_j$  即等于用它们对输入向量空间进行 Voronoi 划分 (参见文献[12]) 后, 各相应子空间的质心。这样, 算法退化为普通 VQ 使用的 LBG 算法<sup>[12]</sup>

### 3.3 SOM 自组织学习算法中的参数自适应

3.2 节中给出了按式 (3-7) 进行迭代计算的自组织学习算法, 式中的  $\alpha(k)$  和  $\Delta(j, j^*(k), k)$  是两个有待选择的函数, 如果后者用式 (3-16) 的形式, 那么有待选择的是步幅函数  $\alpha(k)$  和邻域宽度函数  $\sigma(k)$ 。这是两个很难选择的函数, 如果  $\sigma(k)$  随  $k$  的增加而减小过快则拓扑映射保持这项要求很难满足, 特别当输入向量  $\mathbf{X}$  的分布越不均匀时困难

越大。此外 若  $\mathbf{X}$  的分布是非平稳的, 这时 SOM 的各个权向量  $\mathbf{W}_j$  必须进行自适应调整, 这使得  $\alpha(k)$  和  $\sigma(k)$  的选择更加困难, 因为必须进行“快速”学习来适应环境的变化。文献 [11] 介绍了用 EKF(extended Kalman filter) 的方法来估计 SOM 自组织学习算法中的  $\alpha(k)$  和  $\sigma(k)$  使得 3.2.1 小节所述的 SOM 映射的两项要求俱得以满足。此方法已在第 2 章 2.13.5 小节中介绍, 用于解决 RNN 的参数学习问题, 它也可以用于 MLFN 的参数学习<sup>[19]</sup>。这一小节只介绍用 EKF 估计  $\alpha(k)$  和  $\sigma(k)$  的算法框架, 而不详细讨论 EKF 的细节, 有兴趣的读者可阅文献 [11]、[19]、[20] 和第 2 章给出的相关文献。

### 3.3.1 用 EKF 估计步幅函数

本节讨论的 SOM 有关定义已在 3.2.1 小节中给出, 不再赘述。下面首先定义 EKF 用于 SOM 时的 4 个重要参量, 其他参量在给出算法框架时予以定义。

(1) KF(Kalman Filter) 的“状态”  $\mathbf{W}(k)$

SOM 的  $M$  个权向量  $\mathbf{W}_j = [w_{j1}, \dots, w_{jN}]^T, j = 1 \sim M$  可以构成一个  $MN$  维列向量  $\mathbf{W}$  它是迭代节拍  $k$  的函数, 可以表示为

$$\mathbf{W}(k) = [\mathbf{W}_1^T(k), \mathbf{W}_2^T(k), \dots, \mathbf{W}_M^T(k)]^T \quad (3-23)$$

$\mathbf{W}(k)$  即称为一个 KF 的状态 应按节拍  $k$  不断对其进行估计, 以使得某种目标函数渐近地达到最小值。这实际上是一种动态跟踪过程。

(2) SOM 的邻域函数矩阵  $\mathbf{H}_{jj^*(k)}(k)$

设节拍  $k$  的输入向量为  $\mathbf{X}(k) = [x_1(k), \dots, x_N(k)]^T$ , 前一节拍 SOM 的状态为  $\mathbf{W}(k-1)$  则获胜端  $j^*(k)$  满足式 3-1) 现重写如下:

$$j^*(k) = \arg \min_j \|\mathbf{X}(k) - \mathbf{W}_j(k-1)\| \quad (3-24)$$

如果用式 3-15) 和 (3-16) 计算邻域函数  $\Lambda(j, j^*(k), k)$  (设  $\sigma(k)$  已知) 则可用这些邻域函数值构成一个  $MN \times MN$  维方阵  $\mathbf{H}_{jj^*(k)}(k)$ 。这是一个对角阵 其对角线以外元素皆为 0, 它的前  $N$  个对角元素皆取相同值, 即  $\Lambda(1, j^*(k), k)$  它的第二组的  $N$  个对角元素皆取相同的  $\Lambda(2, j^*(k), k)$  余类推 最后第  $M$  组的  $N$  个对角元素则都取相同的  $\Lambda(M, j^*(k), k)$ 。 $\mathbf{H}_{jj^*(k)}(k)$  即称为邻域函数矩阵。

(3) KF 的“观察”  $\mathbf{Y}(k)$

首先 将 SOM 阵列中每个神经元  $j$  的输出从标量  $y_j$  (见式 3-2)) 改为向量  $\mathbf{Y}_j(k) = [y_{j1}(k), \dots, y_{jN}(k)]^T$  其计算公式是  $\mathbf{Y}_j(k) = \Lambda(j, j^*(k), k) \mathbf{W}_j(k-1)$ 。那么  $M$  个神经元的输出  $\mathbf{Y}_j(k), j = 1 \sim M$  可以构成一个  $MN$  维列向量  $\mathbf{Y}(k)$  即有

$$\mathbf{Y}(k) = [\mathbf{Y}_1^T(k), \mathbf{Y}_2^T(k), \dots, \mathbf{Y}_M^T(k)]^T \quad (3-25)$$

$\mathbf{Y}(k)$  即称为 KF 的观察。根据邻域函数矩阵  $\mathbf{H}_{jj^*(k)}(k)$  的定义, 可以求得 SOM 作为一个 KF 其观察和状态有如下关系 (其中  $\mathbf{N}_Y(k)$  是  $MN$  维观察噪声列向量):

$$\mathbf{Y}(k) = \mathbf{H}_{jj^*(k)}(k) \mathbf{W}(k-1) + \mathbf{N}_Y(k) \quad (3-26)$$

(4) KF 的“输入”  $\mathbf{X}(k)$

已知 SOM 的输入是  $\mathbf{X}(k) = [x_1(k), \dots, x_N(k)]^T$  现在可以构成一个  $MN$  维列向量  $\mathbf{X}(k)$  如下:

$$\mathbf{X}(k) = [\mathbf{X}^T(k), \mathbf{X}^T(k), \dots, \mathbf{X}^T(k)]^T \quad (3-27)$$

$\mathbf{X}(k)$ 即是 KF 的输入。

下面列出由滤波和预测两部分构成的 EKF 迭代计算 每一迭代节拍  $k$  依次由下列两组计算完成，其中有若干新参数将在下面定义。

滤波计算(第 1 组)：

$$\mathbf{K}_a(k) = \mathbf{P}_{a,k-1}(k) \mathbf{H}_{jj}^{T \cdot (k)}(k) [\mathbf{H}_{jj}^{T \cdot (k)}(k) \mathbf{P}_{a,k-1}(k) \mathbf{H}_{jj}^{T \cdot (k)}(k) + \mathbf{Q}_Y(k)]^{-1} \quad (3-28)$$

$$\mathbf{W}_k(k) = \mathbf{W}_{k-1}(k) + \mathbf{K}_a(k) \mathbf{H}_{jj}^{T \cdot (k)}(k) [\mathbf{X}(k) - \mathbf{W}_{k-1}(k)] \quad (3-29)$$

$$\mathbf{P}_{a,k}(k) = \mathbf{P}_{a,k-1}(k) - \mathbf{K}_a(k) \mathbf{H}_{jj}^{T \cdot (k)}(k) \mathbf{P}_{a,k-1}(k) \quad (3-30)$$

预测计算(第 2 组)：

$$\mathbf{W}_k(k+1) = \mathbf{W}_k(k) \quad (3-31)$$

$$\mathbf{P}_{a,k}(k+1) = \mathbf{P}_{a,k}(k) + \mathbf{Q}_W(k) \quad (3-32)$$

对于上列诸式中尚未定义的参量说明如下。 $\mathbf{K}_a(k)$ 是一个  $MN \times MN$  维对角阵，称为增益阵（其意义见下文 3）。 $\mathbf{P}_{a,k-1}(k)$ 也是  $MN \times MN$  维对角阵，它是基于节拍  $k-1$  参数对节拍  $k$  状态  $\mathbf{W}(k)$  进行估计时  $\mathbf{W}(k)$  的协方差阵。 $\mathbf{Q}_Y(k)$ 是  $MN \times MN$  维对角阵，是观察噪声  $\mathbf{N}_Y(k)$  的协方差阵，在计算中其对角元素取恒定的小正数（见下文）。 $\mathbf{W}_k(k)$ 是在节拍  $k$  获得的状态  $\mathbf{W}(k)$  估计， $\mathbf{W}_{k-1}(k)$ 是在节拍  $k-1$  获得的状态  $\mathbf{W}(k)$  估计（它们和  $\mathbf{W}(k)$  一样都是  $MN$  维列向量）。 $\mathbf{P}_{a,k}(k)$ 也是  $MN \times MN$  维对角阵，是基于节拍  $k$  参数对节拍  $k$  状态估计时  $\mathbf{W}(k)$  的协方差阵。 $\mathbf{Q}_W(k)$ 是状态噪声的协方差阵，亦为  $MN \times MN$  维对角阵，其对角元素取恒定小正值（如何选择见下文）。

迭代计算过程描述如下。迭代按节拍  $k = 1, 2, 3 \dots$  进行，进行前需设定下列初值：权向量  $\mathbf{W}_{j,0}(1), j = 1 \sim M$  状态协方差阵  $\mathbf{P}_{a,0}(1)$ 。对于节拍  $k = 1$  在输入  $\hat{\mathbf{X}}(k)$  后，即可用已知的各个  $\mathbf{W}_{j,0}(1)$  求得  $\mathbf{H}_{jj}^{T \cdot (k)}(k)$ 。再根据式 (3-28) 用  $\mathbf{P}_{a,0}(1), \mathbf{H}_{jj}^{T \cdot (k)}(k)$  和  $\mathbf{Q}_Y(k)$  求得  $\mathbf{K}_a(k)$ 。然后根据式 (3-29)，用已知诸变量求得  $\mathbf{W}_1(1)$ 。注意该式中的  $\mathbf{W}_1(1)$  是由  $M$  个  $\mathbf{W}_{j,1}(1)$  构成的  $MN$  维列向量， $\mathbf{W}_0(1)$  是由  $M$  个  $\mathbf{W}_{j,0}(1)$  构成的  $MN$  维列向量。最后用式 (3-30) 求  $\mathbf{P}_{a,1}(1)$ 。以上是第 1 组计算。由第 2 组计算可求得  $\mathbf{W}_1(2)$  和  $\mathbf{P}_{a,1}(2)$ 。对于节拍  $k = 2, 3, 4 \dots$  可按同一方法计算。

对于这一算法作如下几点说明。

(1) 此算法按节拍  $k$  不断对 KF 的状态进行估计和修正  $\mathbf{W}_{k-1}(k)$  是（用前拍参数对本拍状态的）估计  $\mathbf{W}_k(k)$  是修正后的状态。修正目标是减少修正状态与实际状态之差距。在采用有监督学习的情况（如 MLFN 见 2.13.5 小节）其 EKF 算法中可以由训练集的理想输出求得所述的实际状态。对于用无监督学习的 SOM 而言，只有输入  $\hat{\mathbf{X}}(k)$  可资利用。在上述滤波计算的第 2 步（式 3-29）中，即以  $\mathbf{H}_{jj}^{T \cdot (k)}(k) \mathbf{W}_{k-1}(k)$  作为前拍对本拍的估计，以  $\mathbf{H}_{jj}^{T \cdot (k)} \mathbf{X}(k)$  作为本拍实际状态。

(2) 此算法通过式 (3-29) 的计算，使修正状态  $\mathbf{W}_k(k)$  与实际状态之间的差距随  $k$  的增加（按均方意义或概率意义）不断减小<sup>[10,11]</sup>

(3) SOM 自组织学习算法中的步幅函数  $\alpha(k)$  可以由增益阵  $\mathbf{K}_a(k)$  获得。但是与标准算法不同，此处诸  $\mathbf{W}_j(k)$  的分量  $w_{ji}(k), j = 1 \sim M, i = 1 \sim N$  不是取统一的  $\alpha(k)$  而具有各

自的步幅  $\alpha_{ji}(k)$ 。若  $\mathbf{K}_a(k) = \text{diag}\{\mathcal{K}_{11}(k), \dots, \mathcal{K}_{1N}(k), \mathcal{K}_{21}(k), \dots, \mathcal{K}_{2N}(k), \dots, \mathcal{K}_{M1}(k), \dots, \mathcal{K}_{MN}(k)\}$  则有  $\alpha_{ji}(k) = \mathcal{K}_{ji}(k)$ 。

(4) 在各项计算中只有滤波计算的第 1 步 (式 3-28)) 涉及矩阵求逆计算。该矩阵是  $MN \times MN$  维的, 如果  $M$  和  $N$  值较大且该矩阵为非对角阵, 则求逆的计算量很大。幸而, 式中的  $\mathbf{H}_{jj^*(k)}(k)$ 、 $\mathbf{P}_{a,k-1}(k)$  和  $\mathbf{Q}_v(k)$  都是对角阵, 所以只需做对角阵的求逆运算, 计算开销有限。这里假定了  $\mathbf{W}_{k-1}(k)$  的各个分量不相关, 这才使得  $\mathbf{P}_{a,k-1}(k)$  成为对角阵。 $\mathbf{Q}_v(k)$  是对角阵的依据在于假设观察噪声各分量互不相关。

(5) 在计算  $\mathbf{H}_{jj^*(k)}(k)$  时需涉及  $\Delta(j, j^*(k), k)$  的计算, 其中所用的邻域宽度函数  $\sigma(k)$  是由下面 3.3.2 小节中估计  $\sigma(k)$  的 EKF 算法提供的。

(6) 对于平稳输入, 此算法可独立使用, 也可以作为 SOM 的普通 (固定  $a(k)$ ) 函数自组织学习算法的准备算法以 EKF 的结果作为其学习初值。

(7) 对于非平稳输入, 可用此算法提供随环境变化而不断变化的各个权向量  $\mathbf{W}_j(k)$ 。

### 3.3.2 用 EKF 算法求邻域宽度函数 $\sigma(k)$

对  $\sigma(k)$  进行正确估计的目标是使 SOM 实现保持拓扑特性的映射。即 SOM 阵列中两个神经元的几何距离越小则相应权向量的欧氏距离也越小; 反之亦然。问题在于需要有一个可计算的目标函数来作为 SOM 映射拓扑特性保持良好性的量度。已有许多文献给出了关于这一目标函数的定义<sup>[21~28]</sup>, 下面只介绍其中的一种——拓扑积 (topographic product), 它便于进行 EKF 计算<sup>[21,22]</sup>, 其不足处的弥补将最后讨论。

首先对拓扑积进行定义 (关于 SOM 的结构仍按 3.2.1 小节的定义) 其过程如下。

(1) 对于 SOM 阵列中任一神经元  $j$ , 设阵列中与其最接近 (距离最短) 的神经元标号是  $\rho_1(j)$  次接近的标号是  $\rho_2(j)$  依次类推排在距离为  $l$  位的神经元标号是  $\rho_l(j)$ 。如果用式 (3-6) 计算阵列中  $a$  和  $b$  两个神经元之间的距离  $d_{ab}$ , 那么上述这些标号的定义可用下列公式表述:

$$\begin{aligned}\rho_1(j): d_{j\rho_1(j)} &= \min_j d_{jj} \\ \rho_2(j): d_{j\rho_2(j)} &= \min_{j(\rho_1(j) \text{ 除外})} d_{jj}\end{aligned}$$

余可类推。这样  $\rho_1(j), \rho_2(j), \dots, \rho_l(j), \dots$  依次由近及远地排列了神经  $j$  的邻近神经元标号。

(2) 设对任一神经元  $j$  的权  $\mathbf{W}_j$  与其他神经元的权进行比较, 找到与其欧氏距离最小的权; 相应的神经元标号记为  $\varphi_1(j)$  次小者记为  $\varphi_2(j)$  等等。其数学表达式如下:

$$\begin{aligned}\varphi_1(j): \|\mathbf{W}_j - \mathbf{W}_{\varphi_1(j)}\| &= \min_j \|\mathbf{W}_j - \mathbf{W}_j\| \\ \varphi_2(j): \|\mathbf{W}_j - \mathbf{W}_{\varphi_2(j)}\| &= \min_{j(\varphi_1(j) \text{ 除外})} \|\mathbf{W}_j - \mathbf{W}_j\|\end{aligned}$$

余可类推。这样  $\mathbf{W}_{\varphi_1(j)}, \mathbf{W}_{\varphi_2(j)}, \dots$  与  $\mathbf{W}_j$  的欧氏距离按照从小到大的顺序排列。

(3) 如果对于 SOM 阵列中的每一个神经元  $j, \rho_1(j), \rho_2(j), \dots$  和  $\varphi_1(j), \varphi_2(j), \dots$  这两个标号序列完全一致, 那么映射的拓扑保持特性得以完全实现。如果不完全一致, 则可以

用下列  $D_1, \hat{D}_1, D_2, \hat{D}_2$  这四个参数来衡量拓扑保持特性的良度。

$$D_2 = \frac{1}{MM_0} \sum_{j=1}^M \sum_{m=1}^{M_0} \frac{1}{m} \sum_{l=1}^m \lg d_{j\varphi_l(j)} \quad (3-33)$$

$$\hat{D}_2 = \frac{1}{MM_0} \sum_{j=1}^M \sum_{m=1}^{M_0} \frac{1}{m} \sum_{l=1}^m \lg d_{j\hat{\varphi}_l(j)} \quad (3-34)$$

$$D_1 = \frac{1}{MM_0} \sum_{j=1}^M \sum_{m=1}^{M_0} \frac{1}{m} \sum_{l=1}^m \lg \| \mathbf{w}_j - \mathbf{w}_{\rho_l(j)} \| \quad (3-35)$$

$$\hat{D}_1 = \frac{1}{MM_0} \sum_{j=1}^M \sum_{m=1}^{M_0} \frac{1}{m} \sum_{l=1}^m \lg \| \mathbf{w}_j - \mathbf{w}_{\hat{\rho}_l(j)} \| \quad (3-36)$$

注意  $M$  是 SOM 阵列中神经元总数,  $1 \leq M_0 \leq M$ 。  $M_0$  是一个可选择的参数, 它涉及计算拓扑保持的邻域范围, 选择原则将留待本节最后再述。可以看到  $D_2 \geq \hat{D}_2$  当且仅当  $\rho_l(j)$  的排序与  $\varphi_l(j)$  排序一致时 (即拓扑关系保持最佳时)  $D_2 = \hat{D}_2$  二者差距越大 保持越劣。同样  $D_1 \geq \hat{D}_1$  当且仅当  $\rho_l(j)$  与  $\varphi_l(j)$  排序一致时  $D_1 = \hat{D}_1$ ,  $D_1$  越大 拓扑保持越差。这样 可以用  $D_1 - \hat{D}_1$  以及  $D_2 - \hat{D}_2$  作为映射拓扑特性保持良好度的衡量函数, 其值越小良好度越佳。

下面给出以  $D_1$  和  $D_2$  为目标函数的 EKF 迭代算法来估计  $\sigma(k)$ 。与估计  $\alpha(k)$  的 EKF 算法一样, 迭代计算分成滤波计算和预测计算两部分。只是其有关参量 (如状态、观察等) 的定义不同。

在给出迭代计算方程前, 先对若干参量予以定义 (设迭代计算节拍为  $k$ ) 其他参量以后定义。

(1) KF 的状态  $\mathbf{S}(k)$

$\mathbf{S}(k)$  是一个 3 维列向量, 可表示为

$$\mathbf{S}(k) = [D_1(k), D_2(k), \sigma(k)]^T \quad (3-37)$$

$D_1(k), D_2(k)$  分别为节拍  $k$  时按式 (3-35) 和式 (3-33) 计算得到  $\sigma(k)$  为邻域宽度函数。

(2) KF 的观察  $\mathbf{Y}(k)$

$\mathbf{Y}(k)$  是一个 2 维列向量  $\mathbf{Y}(k) = [y_1(k), y_2(k)]^T$ ,  $\mathbf{Y}(k)$  与  $\mathbf{S}(k)$  之间的关系为

$$\mathbf{Y}(k) = \mathbf{C}\mathbf{S}(k) + \mathbf{N}_Y(k), \quad \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad (3-38)$$

其中  $\mathbf{N}_Y(k)$  是一个 2 维观察噪声列向量, 其协方差阵为  $\mathbf{Q}_Y(k)$ 。如果  $\mathbf{N}_Y(k) = \mathbf{0}$  则有  $\mathbf{Y}(k) = [D_1(k), D_2(k)]^T$ 。

(3) KF 的理想观察  $\mathbf{Y}(k) = [\hat{y}_1(k), \hat{y}_2(k)]^T$

设理想的状态是  $\mathbf{S}(k) = [D_1(k), D_2(k), \hat{\sigma}(k)]^T$  其中  $D_1(k), D_2(k)$  由式 (3-36)、(3-34) 计算,  $\hat{\sigma}(k)$  是理想邻域宽度。则理想观察  $\mathbf{Y}(k)$  的计算公式是

$$\mathbf{Y}(k) = \mathbf{C}\mathbf{S}(k) = [D_1(k), D_2(k)]^T \quad (3-39)$$

下面给出由滤波和预测两部分构成的 EKF 迭代计算方程, 每一节拍  $k$  依次完成下列滤波和预测两部分计算,  $k = 1, 2, \dots$ 。

滤波方程包括下列 3 步计算：

$$\mathbf{K}_\sigma(k) = \mathbf{P}_{\sigma,k-1}(k) \mathbf{C}^T [\mathbf{C} \mathbf{P}_{\sigma,k-1}(k) \mathbf{C}^T + \mathbf{Q}_Y(k)]^{-1} \quad (3-40)$$

$$\mathbf{S}_k(k) = \mathbf{S}_{k-1}(k) + \mathbf{K}_\sigma(k) [\mathbf{Y}(k) - \mathbf{C} \mathbf{S}_{k-1}(k)] \quad (3-41)$$

$$\mathbf{P}_{\sigma,k}(k) = \mathbf{P}_{\sigma,k-1}(k) - \mathbf{K}_\sigma(k) \mathbf{C} \mathbf{P}_{\sigma,k-1}(k) \quad (3-42)$$

预测方程包括下列 2 步计算：

$$\mathbf{S}_k(k+1) = \mathbf{S}_k(k) \quad (3-43)$$

$$\mathbf{P}_{\sigma,k}(k+1) = \mathbf{\Psi}(k) \mathbf{P}_{\sigma,k-1}(k) \mathbf{\Psi}^T(k) + \mathbf{Q}_s(k) \quad (3-44)$$

其中  $\mathbf{\Psi}(k)$  是一个  $3 \times 3$  方阵：

$$\mathbf{\Psi}(k) = \begin{bmatrix} 1 & 0 & \mathbf{\Psi}_{13}(k) \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{\Psi}_{13}(k) = \frac{1}{MM_0} \sum_{j=1}^M \sum_{m=1}^{M_0} \frac{1}{m} \sum_{l=1}^m \times$$

$$\frac{-\text{Sgn}[1 - \alpha(k) \Delta(j, \rho_l(j), k)] \alpha(k) \Delta(j, \rho_l(j), k) \|\mathbf{W}_j(k) - \mathbf{W}_{\rho_l(j)}(k)\|^2}{\sigma^3(k) |1 - \alpha(k) \Delta(j, \rho_l(j), k)|} \quad (3-45)$$

对于上列诸式中未定义的参量说明如下。

$\mathbf{K}_\sigma(k)$  是一个  $3 \times 2$  维矩阵，称为增益阵。 $\mathbf{P}_{\sigma,k-1}(k)$  是一个  $3 \times 3$  维方阵，它是基于节拍  $k-1$  的参数对节拍  $k$  的状态  $\mathbf{S}(k)$  进行估计时  $\mathbf{S}(k)$  的协方差阵。 $\mathbf{Q}_Y(k)$  是一个  $2 \times 2$  维对角阵，它是观察噪声  $\mathbf{N}_Y(k)$  的协方差阵，在迭代计算中其对角元素取为固定的小正数（见下文）。 $\mathbf{S}_k(k)$  是节拍  $k$  时对状态  $\mathbf{S}(k)$  的估计， $\mathbf{S}_{k-1}(k)$  则是节拍  $k-1$  时对  $\mathbf{S}(k)$  的估计。它们和  $\mathbf{S}(k)$  一样，都是 3 维列向量， $\mathbf{P}_{\sigma,k}(k)$  也是一个  $3 \times 3$  维方阵，它是基于节拍  $k$  的参数，对节拍  $k$  的状态  $\mathbf{S}(k)$  进行估计时  $\mathbf{S}(k)$  的协方差阵。 $\mathbf{\Psi}(k)$  的第 1 行、第 3 列元素  $\mathbf{\Psi}_{13}(k)$ ，在进行计算时用到节拍  $k$  的诸权向量  $\mathbf{W}_j(k)$ 、 $\rho_l(j)$  的定义已在前面给出， $\Delta(j, \rho_l(j), j)$  用节拍  $k$  的邻域宽度函数  $\sigma(k)$  计算（ $\sigma(k)$  是  $\mathbf{S}_k(k)$  的第 3 个分量，已在式 (3-41) 中求得）式中所用的步幅函数  $\alpha(k)$  用前述的 EKF 算法求得。 $\mathbf{Q}_s(k)$  是一个  $3 \times 3$  维方阵，它是状态噪声（即  $\mathbf{S}(k)$  噪声）的协方差阵，在迭代计算中其元素取小的固定正数值（见下文）。

下面对于迭代计算过程略作说明。计算按节拍  $k = 1, 2, \dots$  进行。首先，在开始迭代计算前需设置若干参量的初值。它们是  $\mathbf{W}_j(0)$ （其中  $j = 1 \sim M$ ）， $\sigma(0)$  和  $\mathbf{P}_{\sigma,0}(1)$ 。用诸  $\mathbf{W}_j(0)$  根据式 (3-35) 和式 (3-33) 可求得  $D_1(0)$  和  $D_2(0)$ ，从而可以算出  $\mathbf{S}_0(1) = [D_1(0), D_2(0), \sigma(0)]^T$ 。其次，开始迭代计算。对于节拍  $k = 1$ ，送进一个输入向量  $\mathbf{X}(k)$ 。先用 3.3.1 小节的算法求得各步幅函数  $\alpha_{ji}(k)$ 。再用  $\sigma(0)$  和诸  $\alpha_{ji}(k)$  按照式 (3-7) 求得  $\mathbf{W}_j(1)$ ， $j = 1 \sim M$ 。接着，用式 (3-40) 和 (3-42) 依次求得  $\mathbf{K}_\sigma(1)$ 、 $\mathbf{S}_1(1)$  和  $\mathbf{P}_{\sigma,1}(1)$ 。然后，用式 (3-43) 和式 (3-44) 求得  $\mathbf{S}_1(2)$  和  $\mathbf{P}_{\sigma,1}(2)$ 。注意  $\mathbf{S}_1(2) = [D_1(1), D_2(1), \sigma(1)]^T$ ，所以可得到  $\sigma(1)$ 。这样，对于节拍  $k = 2$  就可以执行与  $k = 1$  相同的迭代计算。对于  $k = 3, 4, \dots$  可依此类推。



### 3.3.3 参数及初值设置讨论

#### 1. $M_0$ 的选择及输入数据流形 (manifold)

在计算各拓扑积参数  $D_1, D_2$  等 (式 3-33)~(3-36) 时需选择参数  $M_0$ 。如果选择  $M_0 = M$ , 则此种目标函数只适用于线性分布数据流形<sup>[22]</sup>。作为示例 图 3-3 显示了当输入向量为 2 维情况 ( $\mathbf{X} = [x_1, x_2]^T$ ) 下的两种数据分布。其中 (a) 为线性分布数据流形, (b) 为非线性分布数据流形, 每个小圆表示一个输入数据样本。如果用一个 SOM1 维阵列来实现对此输入的映射, 则该直线阵列中各神经元的权  $\mathbf{W}_j = [w_{j1}, w_{j2}]^T$  应该沿 (a) 和 (b) 图中的虚线依次分布。在采用拓扑积准则时, 文献[22] 中业已证明 如果选  $M_0 = M$  就不能使  $\mathbf{W}_j$  做到这一点。为了克服这一困难, 可以选  $M_0 < M$ 。文献[11] 在将 EKF 算法用于 SOM 时, 对于图 3-3(b) 的非线性分布数据流形, 当  $M = 15$  时取  $M_0 = 4$  可实现良好拓扑保持特性。其他的拓扑保持衡量准则尚有拓扑函数等, 可见文献[22]、[23]、[25]、[26]、[27]等。它们可以用于非线性分布数据流形, 但不适宜于 EKF 这一类逐个节拍的自适应计算。

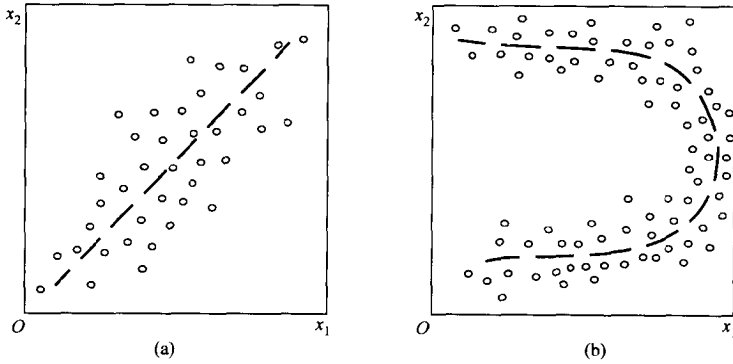


图 3-3 输入为 2 维时的两种数据分布

#### 2. 协方差阵及初值设置

文献[22] 给出了若干模拟实验中所用的初值和协方差阵设置, 可供参考。

对于  $\alpha(k)$  的学习, 有

$$\mathbf{Q}_w(k) = \text{diag}(10^{-2}, 10^{-2}, \dots, 10^{-2})$$

$$\mathbf{Q}_r(k) = \text{diag}(10^{-5}, 10^{-5}, \dots, 10^{-5})$$

$$\mathbf{P}_{\sigma_0}(1) = \text{diag}(10^{-5}, 10^{-5}, \dots, 10^{-5})$$

对于  $\sigma(k)$  的学习, 有

$$\mathbf{Q}_s(k) = \begin{bmatrix} 10^{-10} & 0 & 10^{-10} \\ 0 & 10^{-4} & 0 \\ 0 & 0 & 10^{-10} \end{bmatrix}$$

$$\mathbf{Q}_r(k) = \begin{bmatrix} 10^{-8} & 0 \\ 0 & 10^{-8} \end{bmatrix}$$

## 3.4 学习向量量化——LVQ 和 LVQ2

### 3.4.1 SOM 用于模式识别

前述各节介绍了 SOM 用于聚类、编码等领域时的运行特点及其自组织学习算法。这一小节将介绍 SOM 在模式识别领域的应用。此任务可概述如次。设输入向量  $\mathbf{X}$  属于  $Q$  种模式类别  $C_1, C_2, \dots, C_Q$ 。现在, 若有一个训练集  $\{\mathbf{X}_p, C_p\}, p = 1 \sim P$  其中  $C_p \in \{C_1, C_2, \dots, C_Q\}$ ,  $C_p$  是  $\mathbf{X}_p$  所属之理想类别。SOM 通过此训练集训练以后, 每输入任一  $\mathbf{X}_p$ , 就能在输出端正确给出其类别归属  $\hat{C}_p$ 。当然, 对于训练集外的数据也应有足够好的推广性能。当 SOM 用于此目的时, 首先应选择阵列神经元数  $M$ 。显而易见 若用每个神经元代表一种类别 当  $\mathbf{X}$  属于某种类别时, 与其相应的神经元输出为 1 其他为 0 这样就立即判断出  $\mathbf{X}$  的类别。这意味着  $M$  至少应不小于  $Q$  即  $M = Q$ 。当然, 也可选  $M > Q$  这时可能有多个神经元与一个类别对应, 其中任何一个神经元的输出为 1 即可判断  $\mathbf{X}$  属于该类别。 $M$  选得越大 对  $\mathbf{X}$  赋值空间的划分越细, 从而使分类效果更好。其缺点是使计算和存储量加大。

SOM 用于模式识别时, 分两阶段进行学习。第一阶段用训练集中的  $\mathbf{X}_p$  进行自组织学习, 如 3.2 节所述。第二阶段用  $\{\mathbf{X}_p, C_p\}$  进行有监督学习, 这就是下面将介绍的 LVQ 和 LVQ2。

### 3.4.2 用 SOM 实现模式识别的特点以及与其他模式识别方法的比较

传统的模式识别方法很多, 如贝叶斯 (Bayes) 分类器、KNN (K nearest neighbour) 分类器等。贝叶斯分类器的准则是, 对于任一时刻  $k$  的输入  $\mathbf{X}(k)$  所得识别结果  $C(k)$  应使得后验概率  $p(C(k)/\mathbf{X}(k))$  为最大, 其数学表示式为

$$C(k) = \operatorname{argmax}_{q=1 \sim Q} p(C_q/\mathbf{X}(k)) \quad (3-46)$$

根据贝叶斯公式, 有

$$p(C_q/\mathbf{X}(k)) = \frac{p(\mathbf{X}(k)/C_q)p(C_q)}{p(\mathbf{X}(k))}$$

由于  $p(\mathbf{X}(k))$  项并不随  $C_q$  的不同而改变, 所以为求得使后验概率  $p(C(k)/\mathbf{X}(k))$  为最大的  $C(k)$  作为识别结果, 等效于求下列结果:

$$C(k) = \operatorname{argmax}_{q=1 \sim Q} p(\mathbf{X}(k)/C_q)p(C_q) \quad (3-47)$$

从表面看, 贝叶斯分类器是一种理想分类器, 但是实现起来却相当困难。原因是首先要确定先验概率  $p(\mathbf{X}(k)/C_q)$  的解析表示式 (例如, 常设其为正态分布, 这时即得到熟知的最大似然分类器), 而假设的概率分布与实际概率分布往往并不吻合。其次, 即使概率分布假设得正确, 对其中各项参数 (如正态分布的均值和方差) 的估计也不易准确。这需要大量样本进行统计, 而且往往会出现病态问题。简言之, 像贝叶斯分类器这种基于参数模型的模式识别系统虽有概念清晰、计算方便之利, 而实际操作的效果不一定最佳。与此成

对照的是 KNN分类器，这是一种非参数模型分类器。其原理是对于每个类别  $C_q$  从训练集中取出若干与其对应的输入向量  $\mathbf{X}$  形成一个集合  $\Omega_q, \Omega_q = \{\mathbf{X}, \mathbf{X} \in C_q\}$ 。在识别时 首先计算输入待识别的向量与各  $\Omega_q$  中的向量的欧氏距离，并找出  $K$  个距离最小者（ $K$  一般选为 5 或 6）。以此  $K$  个选出者中属于某个类别最多的作为识别结果。这一方法既无需建立模型又无需估计参数，所以计算十分简单。但是，每个  $\Omega_q$  中必须包含足够多个样本，当此样本数及类别数  $Q$  皆很大时，为完成识别所需的计算量和存储量都是极大的。否则，也不可能有好效果。

SOM 作为模式识别器也是一种非参数模型式识别器，它除具备这类识别器的优点外还能采取自组织 + LVQ 的学习算法，可以使识别所需的存储量和计算量大大下降。此外，一些实验结果表明，其识别效果远优于贝叶斯分类器和 KNN 分类器<sup>[29]</sup>。它的另一个优点是十分便于嵌入 HMM,因此可用于语音识别等复杂的模式识别任务。

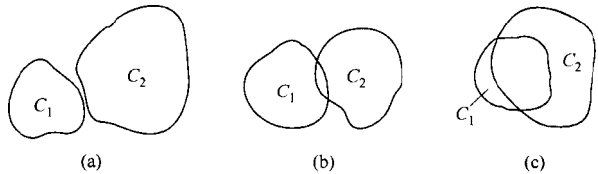


图 3-4  $C_1$  和  $C_2$  之间的相互关系图

模式识别有难易之分，这取决于各个类别  $C_q$  所相应的  $\mathbf{x}$  的赋值空间之间的相互关系。图 3-4 所示是一个只有两种类别（ $C_1$  和  $C_2$ ）情况的示例。其中 (a) 所示为不交迭情况，属易于区分之类；(b) 所示为轻微交迭情况，属于较难区分之类；(c) 所示为严重交迭情况，属于很难区分之类。很多实验结果证明，LVQ 算法可用于易区分的模式识别任务，LVQ2 算法可用于较难或很难区分的任务。

### 3.4.3 LVQ 算法<sup>[30]</sup>

设 SOM 用训练集中的  $\mathbf{X}_p, p = 1 \sim P$ ，按自组织学习算法实施训练以后，阵列中各神经元相应的权向量即可作为 LVQ 算法的初值 它们是  $\mathbf{W}_j(0), j = 1 \sim M$ 。在 LVQ 算法执行前，首先需判断每个神经元  $j$  与  $C_1 \sim C_Q$  中哪个类别相对应。这需要将训练集中的各对  $\{\mathbf{X}_p, C_p\}$  逐对呈示给 SOM 设与  $\mathbf{X}_p$  相应的获胜端是  $j^*(p)$  即  $\mathbf{X}_p$  与  $\mathbf{W}_{j^*(p)}(0)$  的欧氏距离最小，则  $j^*(p)$  应含类别  $C_p$ 。当训练集全部呈示完毕后，可以分两种情况。第一种情况，每个神经元只含一种类别，这时二者对应关系清楚，也无需再施行 LVQ 算法。第二种情况 有若干神经元含有 1 种以上类别，这时如何确定此对应关系呢？最简单的方法是取少数服从多数方案。例如神经元  $j$  含  $C_b$  共 3 次， $C_b$  和  $C_c$  各 1 次，则显然应将其对应类别定为  $C_b$ 。但是，这时每个神经元对于类别的判断将产生一定数量的错误。例如，当神经元  $j$  的输出为 1 时（ $j$  为获胜端），SOM 即判断输入向量属于  $C_a$ ，而实际上它可能属于  $C_b$  或  $C_c$ 。LVQ 的运行正是用来纠正这一类错误的。下面给出 LVQ 算法。

算法按迭代节拍  $k = 0, 1, 2, \dots$  进行。对于每个节拍  $k$ ，依次或随机地从训练集中取

出一对  $\{X(k), C(k)\}$ ,  $C(k)$  是  $X(k)$  所属的理想类别。将  $X(k)$  送入各权为  $W_j(k)$  的 SOM 阵列, 设获胜端是  $j^*(k)$  其相应类别是  $C(k)$ 。这样, 可按下列迭代算法来计算各  $W_j(k+1)$ :

$$W_j(k+1) = \begin{cases} W_j(k) + \alpha(k)[X(k) - W_j(k)], & j = j^*(k), C(k) = \hat{C}(k) \\ W_j(k) - \alpha(k)[X(k) - W_j(k)], & j = j^*(k), C(k) \neq \hat{C}(k) \\ W_j(k), & j \neq j^*(k), j = 1 \sim M \end{cases} \quad (3-48)$$

其中  $\alpha(k)$  为随  $k$  的增加而缓慢下降的步幅函数, 例如可以取  $\alpha(k) = \alpha(0)/k, 0 < \alpha(0) \leq 1$ 。 $\alpha(0)$  应选较小值 例如  $\alpha(0) = 0.05$ 。此外 应令总迭代计算次数  $K$  足够大。例如 当训练集容量为  $P$  时 应令  $K \geq 10P$ 。

LVQ 算法的特点是每个节拍  $k$  只对获胜神经元  $j^*(k)$  的权向量  $W_{j^*(k)}(k)$  进行调整。如果获胜端的类别与理想类别一致, 则令  $W_{j^*(k)}(k)$  朝趋向于  $X(k)$  的方向调整; 反之, 则朝相反方向调整。

### 3.4.4 LVQ2 算法<sup>[31]</sup>

LVQ2 仍以 SOM 自组织学习算法为基础, 其权向量初值设置方法与 LVQ 相同。当迭代按节拍  $k = 0, 1, 2, \dots$  进行时, 每个节拍仍取出一对  $\{X(k), \hat{C}(k)\}$ 。 $X(k)$  送入 SOM 后, 可以有一个获胜端  $j^*(k)$ , 其相应类别为  $C(k)$ , 还有一个次胜端  $\gamma^*(k)$ , 其相应类别为  $\zeta(k)$ 。获胜端的权向量  $W_{j^*(k)}(k)$  和次胜端权向量  $W_{\gamma^*(k)}(k)$  满足下列关系:

$$\|W_{j^*(k)}(k) - X(k)\| < \|W_{\gamma^*(k)}(k) - X(k)\| < \|W_j(k) - X(k)\|, \quad j \neq j^*(k), j \neq \gamma^*(k) \quad (3-49)$$

另外 在  $W_{j^*(k)}(k)$  和  $W_{\gamma^*(k)}(k)$  之间设置一个域  $D(k)$  其定义如下:

$$D(k) = \{X(k) : \|X(k) - X_a\| \leq d \text{ 且 } \|X_a - W_{j^*(k)}(k)\| = \|X_a - W_{\gamma^*(k)}(k)\|\} \quad (3-50)$$

其中  $d$  是一个小正常数。 $D(k)$  是  $W_{j^*(k)}(k)$  和  $W_{\gamma^*(k)}(k)$  之间的分割面 (此面上任一向量  $X_a$  与这两个权向量等距) 附近的一个窄域。图 3-5 给出了 2 维空间中  $D(k)$  的示例。

这样, 可以按下式进行迭代计算:

$$W_j(k+1) = \begin{cases} W_j(k), & j = j^*(k) \text{ 且 } C(k) = \hat{C}(k) \\ W_j(k) + \alpha(k)\{X(k) - W_j(k)\}, & j = \gamma^*(k) \text{ 且 } \\ & C(k) \neq \hat{C}(k), \zeta(k) = \hat{C}(k), X(k) \in D(k) \\ W_j(k) - \alpha(k)\{X(k) - W_j(k)\}, & j = j^*(k) \text{ 且 } \\ & C(k) \neq \hat{C}(k), \zeta(k) = \hat{C}(k), X(k) \in D(k) \\ W_j(k), & \text{其他所有情况} \end{cases} \quad (3-51)$$

其中  $\alpha(k)$  为步幅函数, 其要求与 LVQ 算法一致。此算法的特点是, 只有当第 1 获胜者的类别与理想类别不一致, 而与其最接近的第 2 获胜者的类别与理想类别一致, 且  $X(k)$  位于此二者分割面附近的窄区域中时, 才对二者都进行调整。调整的方向是使  $W_{j^*(k)}(k)$  远离

The diagram shows a vertical crack in a plate. The crack has a width of  $2d$ . The plate has a thickness of  $D(k)$ . The wave speed in the plate is  $W_{f(k)}(k)$  and in the crack is  $W_{\gamma(k)}(k)$ . A cross-section is indicated by a dashed line.

图 3-5 2维空间中  $D(k)$  的示例

LVQ 一致。对每个节拍  $k$  仍取一对条件仍与  $\{\mathbf{X}(k), C(k)\}$ 。  $\mathbf{X}(k)$  输入 SOM 后,  $j^*(k)$  和  $\gamma^*(k)$  分别

 $\zeta(k)$ 。这时可按下列诸式进行
$$\mathbf{W}_j(k+1) = \begin{cases} \mathbf{W}_j(k) + \alpha(k)[\mathbf{X}(k) - \mathbf{W}_j(k)], & j = j^*(k) \\ \mathbf{W}_j(k) - \alpha(k)[\mathbf{X}(k) - \mathbf{W}_j(k)], & j = \gamma^*(k) \text{ 且} \\ \quad \|\mathbf{X}(k) - \mathbf{W}_{j^*(k)}(k)\| - \|\mathbf{X}(k) - \mathbf{W}_{\gamma^*(k)}(k)\| \leq \theta \\ \mathbf{W}_j(k), & j \neq j^*(k) \text{ 或 } \gamma^*(k) \text{ 及其他} \end{cases} \quad (3-52)$$

(2) 若  $C(k) \neq C(k)$  且  $\zeta(k) = C(k)$  则

$$\mathbf{W}_j(k+1) = \begin{cases} \mathbf{W}_j(k) + \alpha(k)[\mathbf{X}(k) - \mathbf{W}_j(k)], & j = \gamma^*(k) \\ \mathbf{W}_j(k) - \alpha(k)[\mathbf{X}(k) - \mathbf{W}_j(k)], & j = j^*(k) \\ \mathbf{W}_j(k), & j \neq j^*(k) \text{ 或 } \gamma^*(k) \end{cases} \quad (3-53)$$

$$\mathbf{W}_j(k+1) = \begin{cases} \mathbf{W}_j(k) + \alpha(k)[\mathbf{X}(k) - \mathbf{W}_j(k)], & j = l(k) \\ \mathbf{W}_j(k), & j \neq l(k) \end{cases} \quad (3-54)$$

其中  $l(k)$  是与类别  $C(k)$  相应的神经元编号。以上诸式中步幅函数  $\alpha(k)$  的选择原则与 LVQ—致。此修正 LVQ2 算法较前述的 LVQ2 算法略简单, 在若干模拟实验中二者的效果相仿。

与其他人工神经网络相似,在实际应用 SOM 时会遇到以下三方面问题。第一,输入信号的选择及其预处理。第二,SOM 的参数选择(关键是阵列的神经元数  $M$ )。第三,如何与其他算法相结合。下面将结合各项具体应用讨论这些问题。SOM 的应用很多,这里只介绍下列几方面:系统状态辨识、控制和故障诊断、图像处理、语音识别、通信。文献[2]给出了截至 1996 年的有关 SOM 应用的参考文献 197 篇,可供进一步查阅。

### 3.5.1 SOM 用于系统状态辨识、控制和故障诊断

一个实际系统往往非常复杂而且是非线性的，因此通过建立严格的数学模型并根据若干对系统的测量（观察）值来确定系统的状态（此状态本身也是一个多维向量），并进一步实施对系统的控制以及预测和发现系统的故障等，即便不是绝对不可能，也是极为困难的。SOM在解决这个难题上提供了一种新途径。如果将系统的各个测量值（设有  $N$  个）形成一个  $N$  维列向量  $\mathbf{X}$ ，则可将它作为一个 SOM 的输入。如果能采集到各种可能的  $\mathbf{X}$  作为训练集，则 SOM 通过自组织学习可以对于任何输入  $\mathbf{X}$  产生一个获胜神经元标号，它在 SOM 的 2 维阵列中有一确定位置点。这个位置点与实际系统的某一状态点（更确切说是一族互相很接近的状态）相互对应。由于 SOM 实现的是保持拓扑特性的映射，因此阵列中较邻近的位置点所对应的系统状态也应是比较接近的。这就为“系统状态视觉化”和“系统行为视觉化”提供了一种非常强有力的工具。图 3-6 给出了一个说明示例，其中黑线包围的方形表示一个 SOM 2 维阵列。图中  $a$  和  $b$  表示与不同时刻输入  $\mathbf{X}$  相应的两个获胜神经元，也就是对应两种系统的状态。 $a$  和  $b$  之间的黑色曲线，表示系统从  $a$  相应的状态经过一段时间运行达到  $b$  所相应状态时所历经的状态轨迹，也就是这一段时间中系统的行为。利用 SOM 阵列所提供的每一离散时刻的输入  $\mathbf{X}$  所对应的获胜端（状态）及其运行轨迹可以帮助一个系统（小至一台机器，大至整个工厂）的监控者做到以下几点：使系统的操纵者或监控者有可能在视觉上跟踪一个系统状态的运行进程，即其行为。根据系统状态的运行轨迹，可以估计未来状态的走向，即预测未来的状态。对于系统现在状态或预测的未来状态是否反常作出判断，即进行故障诊断或进行故障预测。例如，图 3-6 所示阵列中封闭曲线  $e$  所包围的区域为正常区，其外为不正常区，则可根据这一界限进行故障诊断和预测。利用此视觉化的状态对系统进行控制，使其达到某种特定状态或者尽量远离不安全区。下面举一个实际的示例。

设有一大功率变压器，它的 8 个监控参数构成了 SOM 的 8 维输入列向量  $\mathbf{X}$ 。这 8 个参数是：负载电流，线圈温度，平均油温，变压器顶部油温，用于控制负载的抽头位置调节器

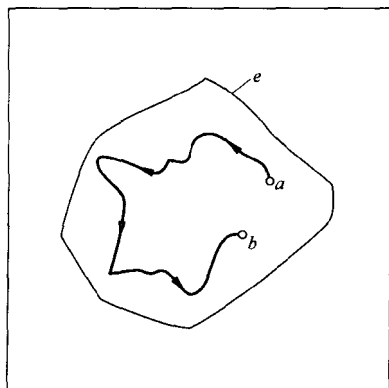


图 3-6 SOM 用于系统状态确定的一个示例

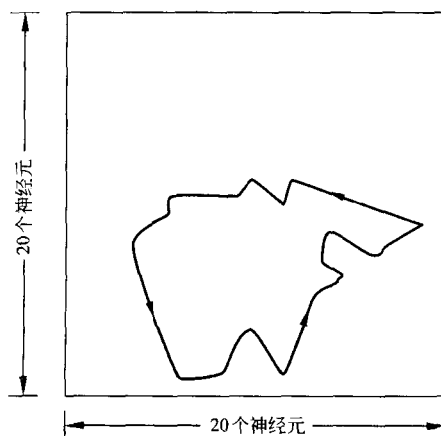


图 3-7 获胜端轨迹

的设定点，电压，氢含量读数，变压器所处房间的室温。这些参数需进行规格化处理，以使它们具有相同的取值范围或具有相同的方差和 0 均值，才成为输入 SOM 的  $X$ 。SOM 阵列含  $M = 400(20 \times 20)$  个神经元，经过自组织学习算法训练后，形成相应的 400 个 8 维权向量  $W_j, j = 1 \sim 400$ 。图 3-7 所示的是这个 SOM 阵列的学习完成后，在 24 小时的实际运行过程中记录到的获胜端轨迹，大致构成一个封闭环路。通过比较每日的运行轨迹，可以发现每天的运行路径大致相同。通过记录每日各时刻 SOM 阵列中获胜端所处位置，即相应于该变压器所处的某种实际状态，从而可以对其状态进行监控并作出故障诊断或预测。

在实施故障诊断或预测时，可以采取两种方案。第一种方案是训练集中的所有  $X$  均取自系统正常运行的状态下，并且用此训练集对 SOM 阵列进行自组织学习式训练。训练完成后，在实际运行中计算每一时刻的  $X$  与阵列中各权向量的欧氏距离，并求出其最小距离。若此最小者低于某一预置阈值，说明系统处于正常状态；若略超过，则处于警戒区；若超过较多，则处于危险区或已出现故障。这一方案是基于充分搜集到各种可能出现的正常状态下的  $x$  作为训练集的。第二种方案是训练集中既包含正常状态的  $X$  又包含不正常（故障）状态的  $x$  以及二者之间的危险区。这样 SOM 阵列在经过学习后，即可在阵列中分出哪些神经元属正常状态，哪些属故障状态及危险状态。这一方案的缺陷在于，当一个实际运行系统较大，较复杂时很难搜集到后两种情况的  $x$ 。作为一种弥补，可以采用人工模拟的方法来形成不正常状态下的  $x$ 。人工模拟可以考虑到实际故障发生的部位和方式来制造  $x$ 。下面列举其中 5 种可能的情况。① 8 个监控信号测试线中，任何一个断线造成的该信号丢失。② 由于严重干扰，造成 8 个信号中任何一个产生大量突发性改变。③ 由于 8 个信号测试设备的机械故障所造成的信号扰动（jamming）。④ 由于测试设备老化等原因造成测量值的慢漂移。⑤ 由于控制机构失灵造成某个恒定信号突变。

SOM 还可以用来进行故障分析，即发现故障后确定故障发生在哪一个部位。这时可以用另外一个 SOM 阵列来做到这一点。此外，如前文已指出，SOM 阵列的系统状态视觉化还可以用来对系统状态进行控制，使之远离故障区或危险区并且达到具有良好运行状态的区域，从而达到质量控制的目标。对于各种大的、复杂的、高度非线性的系统，诸如化工企业、制造业（如芯片制造）经济系统等等，SOM 阵列除上述各项外还可用于过程状态分析和稳定性分析等各种情况。还有一些典型的应用，如动力工程（power engineering），涉及的问题如：负载预测、系统稳定性、部分失效诊断、变压器故障诊断、用电高峰发生时间等<sup>[2]</sup>。还有造纸工业中的应用，如纸质量监督、造纸机故障判断等<sup>[2]</sup>。

SOM 阵列在应用中有很多理论问题被提出来并部分得到了解决，如：大规模诊断问题<sup>[33]</sup>、系统状态辨识<sup>[34]</sup>、故障诊断<sup>[35,36]</sup> 以及工厂故障症状（Symptom）检测<sup>[37]</sup> 和机器振动诊断<sup>[38]</sup> 等。

### 3.5.2 SOM 用于图像信号处理

下面仅举计算机视觉和纹理分析及分类作为例子。

#### 1. 计算机视觉

这是人工神经网络具有很大潜在运用价值的领域之一。在工业自动化和现代化商业

中都将发挥 SOM 的作用，诸如依靠视觉的质量控制、机器人、医学图像处理、遥感信息处理、文件处理和靶 (target) 识别等等。计算机视觉系统可由下列几步操作构成：①对输入的数字化图像进行预处理并分割成若干内部存在关联 (relevant) 的域；②提取出各分割域或目标的特征；③依据各分割域的特征作出分类或解释。常用的特征提取和分类方法是基于模型的方法<sup>[39]</sup>。然而许多目标具有非刚性 (nonrigid) 变换而很难建立明确的模型。如手写字、人脸、自然纹理、材料表面缺陷等都存在此类问题。因此，研究者将希望寄予人工神经网络，特别是用来解决上述第 ③ 项的特征提取任务。对于采用有监督学习的 MLFN 这一类网络，由于其内部参数很多，需要有足够数量的 (输入、理想输出) 样本对构成的训练集，才能使网络的学习具有统计可靠性，这要求训练集容量非常大 (见第 2 章)。SOM 用于计算机视觉的特征提取时提供了一种可能更有利的途径。因为 SOM 采用无监督学习算法，所以无需对输入图像进行预先分类以形成理想输出。经过 SOM 学习后，提供的是数量有限的标记信息作为第 ③ 项分类系统的输入。这种标记信息是原始图像经过高度压缩后的一种表示，因此完成第 ③ 项任务时只需中等数量的样本构成的训练集就足够了。当然，对于使用 SOM 来完成项目 ③ 的特征提取任务时，项目 ① 的预处理仍是完全必要的，以实现预特征提取。这依赖于所处理图像的自身特点进行适当的变换。例如，小波变换、边缘检测、色彩、多谱信息以及基于灰度级共生的纹理测度等<sup>[2]</sup>

下面给出一个 SOM 用于人脸识别的实例<sup>[40,41]</sup> 整个识别系统由 3 部分组成 分别完成上述的 3 步操作 (即预处理、特征提取及分类/识别) 如图 3-8 所示。下面分别介绍它们的工作原理。

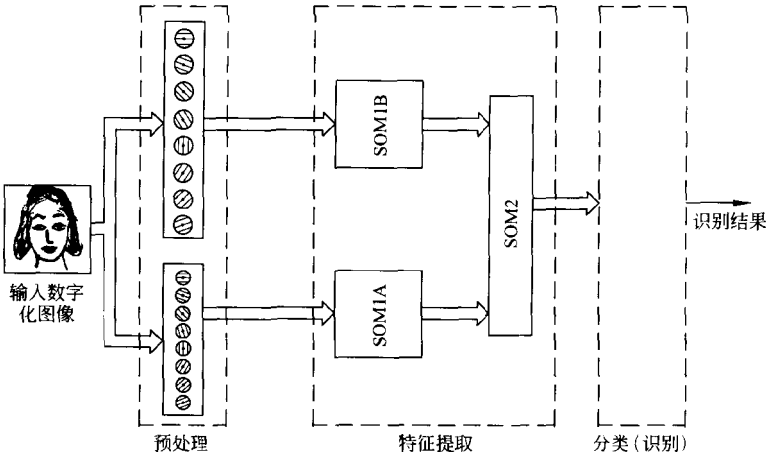


图 3-8 一个用于人脸识别的 SOM 实例

(1) 预处理部分

输入信号是一个  $128 \times 128$  的数字化图像 (即含 16384 个像素) 如果以整个图像的左下角为坐标原点，横轴和纵轴分别用  $x_1$  和  $x_2$  表示，那么每一个像素的位置可以用一对坐标  $(x_1, x_2)$  表示， $x_1, x_2 = 0, 1, \dots, 127$ 。每个像素的灰度级用  $H(x_1, x_2)$  表示。这样，一帧完整的图像即可用所有像素的灰度级来描述。为简化表示，设  $\mathbf{X} = [x_1, x_2]^T$ ， $\mathbf{X}$  的所有取值域为  $D$  则一帧图像可表为  $H(\mathbf{X})$ ， $\mathbf{X} \in D$ 。在预处理部分要做的是对  $H(\mathbf{X})$  进行某种变



换,以提取出各个像素点附近图形中与方向有关(orient-dependent)以及与(空间)频率有关的信息(例如与边缘有关的信息)。为此可以采用2维自相似小波变换(2D-self-similar wavelet)或称为Gabor变换<sup>[40~42]</sup>。对于图形中任何一个像素点 $\mathbf{X}_0$ 的变换式如下:

$$F(f_k, \theta, \mathbf{X}_0) = \int_{\mathbf{X} \in D} \Psi(f_k, \theta, \mathbf{X} - \mathbf{X}_0) H(\mathbf{X}) d\mathbf{X} \quad (3-55)$$

$$\Psi(f_k, \theta, \mathbf{X}) = \exp \left\{ j(\mathbf{f}_{k,\theta}^T \mathbf{X}) - \frac{\|\mathbf{f}_{k,\theta}\|^2 \|\mathbf{X}\|^2}{2\sigma^2} \right\} \quad (3-56)$$

式中 $\mathbf{f}_{k,\theta} = [f_k \cos \theta, f_k \sin \theta]^T$ 。 $f_k$ 可取值为 $f_k = \pi/2^k, k = 0, 1, 2, \dots$ 。 $k$ 值越小时相应的空间频率越高,即分辨率越高。 $\theta$ 决定变换敏感的方向,在本例中取8种方向,因此 $\theta$ 可取值为 $\theta = \theta_l = \frac{\pi}{8}l, l = 0, 1, \dots, 7$ 。 $\sigma$ 决定此变换所取的高斯型空间窗宽度,在本例中取 $\sigma = 50$ 。

在实际运行时,为了求得不同 $k$ 取值的 $F(f_k, \theta, \mathbf{X}_0)$ 只需用同一个 $\Psi(\cdot)$ 函数,即 $\Psi(f_0, \theta, \mathbf{X})$ 。在求 $F(f_0, \theta, \mathbf{X}_0)$ 时可以直接将其代入式(3-55)并且用原始的 $128 \times 128$ 图像相应的 $H(\mathbf{X})$ 来进行计算。在求 $F(f_1, \theta, \mathbf{X}_0)$ 时仍然用此 $\Psi(f_0, \theta, \mathbf{X})$ 而 $H(\mathbf{X})$ 改为由原图像经过降抽样速率处理后形成的图像替代,即后者的 $H(0,0)$ 、 $H(0,1)$ 、 $H(1,0)$ 和 $H(1,1)$ 这4个像素灰度值为前者这4个像素灰度值的平均值。其他像素依此类推。对于求 $F(f_2, \theta, \mathbf{X}_0)$ 等,也可照此用降抽样速率的方法进行。 $\Psi(f_k, \theta, \mathbf{X})$ 称为Gabor变换的核(kernel)。

在这里介绍的人脸识别系统中,取 $k = 0$ 和1(即高和低)两种频率分辨率和8种空间分辨率 $\theta_l, l = 0 \sim 7$ 。这样可以形成两个8维列向量作为预处理级的输出。与高频率分辨率相应的8维列向量的8个分量是 $|F(f_0, \theta_l, \mathbf{X}_0)|, l = 0 \sim 7$ 与低频率分辨率相应的8维向量的8个分量是 $|F(f_1, \theta_l, \mathbf{X}_0)|, l = 0 \sim 7$ 。 $|\cdot|$ 表示取复值函数的模。这样,对于任一原始图像的每个像素 $\mathbf{X}_0$ (共有 $128 \times 128$ 个)各有两个8维向量。

## (2) 特征提取部分

特征提取用2级SOM完成。采用多级SOM(记为MSOM)来替代单级SOM的原因主要是单级SOM只能对输入向量空间实现凸分割,而2级SOM则可以实现包括非凸分割在内的任意分割,这可以参见MLFN的类似分析(见文献[3],2.4节)。详情见文献[40]。第1级SOM含SOM1A和SOM1B两个阵列,每个阵列由 $10 \times 10$ 个神经元构成,其输入分别是高和低频率分辨率8维列向量。每个阵列的输出是其获胜端的纵、横坐标,例如:2,7。这样两个阵列的输出构成一个4维列向量,例如 $[2, 7, 5, 3]^T$ 。这就是第2级SOM(用SOM2表示)的输入。SOM2是一个1维阵列,由100个神经元构成,其输出即是其获胜端在1维阵列中的位置标号,例如79。2级SOM的训练集取自不同的输入图像,其中包含男女老少不同的人脸、动物(如山魃)、建筑物风景、卡通片等等。每帧图像提供 $128 \times 128$ 对8维列向量作为训练样本。训练时,通过自组织学习算法首先确定SOM1A和SOM1B各自的100个8维权向量,然后再用自组织算法确定SOM2的100个4维权向量。

## (3) 分类识别部分

在识别时每一帧人脸图像共含 $128 \times 128$ 个像素,经预处理后每个像素变换为两个8维列向量,再经特征提取后产生一个特征标号 $i^*$ 。其取值范围为 $0 \sim 99$ 。这样共有 $128 \times 128 = 16384$ 个特征标号。因此,如果每一像素位置可表示为 $\mathbf{r} = [r_1, r_2]^T$ 那么任何一帧

人脸图像可表征为

$i^*(\mathbf{r}) \in [0, 99]$ ,  $\mathbf{r} = [r_1, r_2]^T$ ,  $r_1, r_2 = 0 \sim 127$  这就是分类（识别）部分的输入。将此输入首先转换为一个用高斯窗加权平滑的获胜编号直方图  $\phi_i$ ,  $i = 0 \sim 99$ , 其定义如下：

$$\phi_i = \sum_{\mathbf{r}: i^*(\mathbf{r})=i} \Omega(\mathbf{r}_0 - \mathbf{r}), \quad i = 0 \sim 99$$

其中

$$\Omega(\mathbf{r}_0 - \mathbf{r}) = \exp \left\{ -\frac{\|\mathbf{r}_0 - \mathbf{r}\|^2}{2R^2} \right\}$$

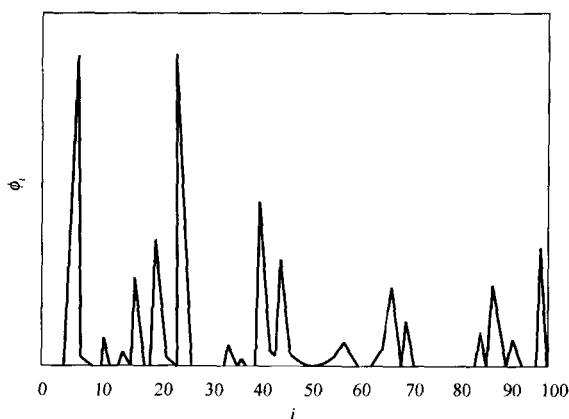


图 3-9 人脸的  $\phi_i \sim i$  示例

式中

$\mathbf{r}_0$  是一帧图像的中心。在本例中取  $R = 50$ ,  $\mathbf{r}_0 = [64, 64]^T$ 。图 3-9 给出了一张人脸的  $\phi_i \sim i$  示例。

不同人的脸具有不同的  $\phi_i \sim i$  直方图，因此可以首先将每个有关人脸的图像输入后，记录下相应的直方图。在识别时，将待识别人脸的直方图与已记录的直方图比较并取其最相似者作为识别的结果。所采取的识别技巧可参阅文献[40]。该文献报道，在训练时取了 17 个人脸的图像，在识别时，这些人脸图像的摄取方向、尺度作中等程度变化且有非刚性变化（例如不同的表情）时，仍可取得良好的识别结果。

## 2. 纹理分析和分类

纹理分析是图像分析的一个重要部分。其作用之一是从一帧图像中切割出具有相似特征从而有内在含意的某个部分，另一个作用是在工业中实现产品质量的视觉监督（如木材、纸张的质量），纹理的确切定义很难给出，但可从实体会其含意，如绒毛、砖墙、草地、水面等等。从工程角度看，纹理分析和分类由下列 4 步操作构成：图像获取和预处理，特征提取，聚类，分类。其中、两项可由 SOM 用自组织学习算法完成<sup>[43,44]</sup>。为了改善④项的分类效果，还可以进一步采用 LVQ 算法<sup>[45,46]</sup>。

上述①、两项对于实现成功的纹理分析和分类特别重要。其中应特别予以关注的有以下几点。第一，在训练和识别时，图像的光源强度及照明条件应保持一致。如二者不一致，则应通过照度直方图的均衡达到光强和对比度的归一化<sup>[47]</sup>。第二，应充分利用有关领域

的专家知识来选取纹理特征<sup>[47,48,49,50]</sup>。第三，像素对之间的统计特性包含了有关纹理的重要统计信息。为描述此关系可以在任一图像窗口中求得共生矩阵（co-occurrence matrix） $M_d(i, j)$ 。其定义是：设每一像素有  $L$  个灰度级，记为  $h_i, i = 1 \sim L$ ，则  $M_d(i, j)$  表示在选择的图像窗口内距离为  $d$  的任意两个像素中第一个为  $h_i$  且第二个为  $h_j$  的情况发生的次数。可以用此  $M_d(i, j)$  或其统计值作为纹理的重要特征<sup>[51,52,53]</sup>。

上述 ③、④ 两项也可以用其他统计（具有参数模型）识别方法或其他人工神经网络。但是由于特征的统计分布往往是非对称的，前者的效果不佳。而后者由于采用有监督的学习算法，所以训练集的容量要求非常大。相比之下，SOM 可以在训练集较小时取得良好结果，所以是一种较好方案<sup>[43,44]</sup>。

### 3.5.3 SOM 用于语音识别

这里介绍一个用于芬兰语的不受限连续语音（特定人）识别系统，目标是实现听写机。整个识别系统由 4 部分构成：①语音预处理，将每一帧输入语音转换为由 20 个听觉倒谱分量构成的特征向量；②用 SOM 将每一帧语音的特征向量转换为似音素（quasiphoneme）；③将似音素流分割为音素段（用 HMM）；④后处理部分，使用词法和文法将音素流转换为书面文字。下面简要描述这几部分的功能。

#### 1. 语音预处理部分

语音采样率为 12.8kHz，一帧语音的长度（帧长）为 20ms（即一帧含 256 个样点）帧移为 10ms。每帧语音用 Hamming 窗加权后作 256 点 FFT 变换，得到 128 个在 0~6.4kHz 范围内均匀分布的对数功率谱分量（分量之间间隔为 50Hz）。然后，将其转换为听觉加权谱分量  $x_k, k = 1 \sim 22$ 。转换方法是将 0~1kHz 均匀（线性）分割为 10 个带，将 1kHz~5kHz 按取对数后均匀分割为 12 个带。 $x_k$  即等于按序编号的带内各前述功率谱分量之和。最后用离散余弦变换将  $x_k$  转换为听觉加权倒谱  $C_i$ ，即有

$$C_i = \sum_{k=1}^{22} x_k \cos \left[ i \left( k - \frac{1}{2} \right) \frac{\pi}{22} \right], \quad i = 1 \sim 20$$

$C_1 \sim C_{20}$  即构成送往下一部分的特征向量。为纳入各语音帧之间的时间相关性，可以将 3 至 7 个特征向量并成一个大向量作为送到下一部分的信号。

#### 2. 用 SOM 实现似音素分类

由于此处需实现的是分类功能而不只是聚类功能，所以可采用自组织 SOM + LVQ 的方案，前文已述及并可参阅文献<sup>[54]</sup>、<sup>[55]</sup>。另一种方案是采用有监督的 SOM 方案<sup>[56]</sup>。这里只介绍这一种。设前一部分送来的特征列向量是  $\mathbf{X}_s$ 。假设共有  $Q$  种似音素，且已知每一帧  $\mathbf{X}_s$  所对应的似音素类别，那么可以相应地形成一个  $Q$  维列向量  $\mathbf{x}_u$ 。设所属类别为  $i$ ，那么  $\mathbf{x}_u$  的第  $i$  分量为 1 而其他分量为 0。这样可以用一扩大的列向量  $\mathbf{X} = [\mathbf{X}_s^T, \epsilon \mathbf{x}_u^T]^T$  作为 SOM 的输入向量。相应地，SOM 各神经元  $j$  的权  $\mathbf{w}_j = [\mathbf{w}_{js}^T, \mathbf{w}_{ju}^T]^T$ 。 $\epsilon$  是一个小正数，例如取  $\epsilon = 0.1$ 。SOM 仍取自组织学习算法。学习完成后，进入识别阶段时，只用  $\mathbf{X}_s$  和  $\mathbf{w}_{js}$  来决定获胜者以给出识别结果。采取此方案可使分类精度明显提高，不亚于 LVQ。

SOM 用于似音素识别时，阵列神经元数  $M$  必须足够大，在这个单说话人的芬兰语识

别系统中取  $M = 200 \sim 300$ 。为了改善识别精度,  $M$  还需要进一步扩大而且输入特征向量应包含更多语音帧(见本小节中的(1))。甚至, 对于平稳段和过渡段, 音素采用不同的 SOM 或不同的 LVQ)<sup>[2]</sup>。

$W_j$  的  $W_{ju}$  部分可以作为神经元  $j$  的识别结果是否可靠的标志。在学习过程中, 如果与神经元  $j$  相应的  $W_{ju}$  中只有 1 个分量接近于  $\varepsilon$  而其他分量很接近 0 则表明该神经元能持续给出正确结果。相反, 如果  $W_{ju}$  中有多个较大的分量, 这表明给出结果参差较大(这往往发生于分类边界处)。对于后一种情况, 应将该神经元去除, 这时能略改善识别结果。

### 3. 将似音素流分割为音素流和将音素流转换为词和句

一个似音素的长度为 10ms, 而一个音素的长度为 40ms ~ 400ms。这两部分转换的工具是 HMM, 这部分内容可在其他文献中找到<sup>[12]</sup> 此处不赘述。

## 3.5.4 SOM 用于通信

SOM 在通信中的可应用的方面很多, 诸如数字通信中的信号检测与均衡、同频道干扰的抑制、语音编码等。这里只介绍 SOM 在抗干扰图像编码中的应用。采用 VQ 向量量化) 对一帧图像进行编码 → 传输(或存储) → 解码, 是图像压缩和编码的一种有效方法, 得到广泛应用。由于 SOM 可作为 VQ 实现的一种方案, 由其形成的 VQ 码本(codebook) 性能与其他方案比较受到关注。研究结果表明, SOM 方案与其他方案(如 LBG 算法<sup>[12]</sup>) 所得到的码本在性能上差别不大<sup>[57, 58, 59]</sup>。但是, 普通 VQ 码本的各标号是无序的, 如果在传输中出错, 则收端译码得到的图像与发端编码的图像大相径庭。而 SOM 产生的码本中各标号是有序的——相邻标号对应着相似的输入图像。这样, 即使在传输中出错, 收发两端的图像也不会有太大差异。因此, 用 SOM 实现图像的 VQ 编译码较之普通 VQ 编译码具有更好的抗干扰能力。下面给出一个模拟实验的例子。

首先, 每一帧待编码的图像分割成  $4 \times 4$  的互不覆盖图像块, 每一块中含 16 个像素, 相应的 16 个灰度级形成一个 16 维向量  $\mathbf{X}$ 。训练集由 28 帧图像中所有图像块构成, 测试集则是另一帧未参与训练的图像。SOM 阵列以  $\mathbf{X}$  为输入, 经过自组织学习可以确定阵列中每一个神经元的权向量。在工作时每输入一个  $\mathbf{X}$ , SOM 阵列有一获胜端, 将此获胜端标号以数字方式传到收端, 收端即将该标号的权向量(收发两端存储完全相同的两套权向量) 作为译码输出。在传输时每个图像块用 9 个比特来传输, 为此可以采用两种方案。一种方案是采用 8 电平脉幅调制(PAM) 每个 PAM 信号可传送 3 个比特( $\log_2 8$ ) 所以传送一个图像块需用 3 个 PAM 信号。另一方案是采用只有 0 和 1 两种电平的二进制信号, 这时每个信号只能传送 1 个比特( $\log_2 2$ )。为适应前一种方案, 可以采用一个 3 维 SOM 阵列, 阵列的边长是 8 个神经元, 所以阵列中共含  $8^3 = 512$  个神经元。传送获胜端标号时每一维与一个 PAM 信号对应, 各维中按序编号的 8 个神经元与 PAM 的 8 个电平对应。在传送过程中, 由于存在噪声与干扰, 每个 PAM 信号的发送电平在收端有可能错成另一个相邻电平(由于噪声一般较小, 所以只考虑这种错误, 而不考虑两个及两个以上电平间隔的错误)。假设从一个电平错误到另一个电平的概率为  $p$ , 则收端将获胜端标号译错的概率  $P$  可由下式计算得到:  $P = 1 - [1 - (7/4)p]^3$ 。当  $p = 0.1$  时,  $P = 0.438$ 。这说明译错概率是很大的。对

于第二种方案，则可以采用一个 9 维 SOM 阵列，阵列的边长是 2 个神经元 所以阵列中共含  $2^9 = 512$  个神经元。传送获胜端标号时每一维与一个二进制信号对应。各维中一个神经元与二进制信号的 0 对应 另一个与 1 对应。用这两种方案所做的模拟实验结果表明，当噪声较大时（前一方案的  $p = 0.1$  相应于较大噪声），它们所恢复的图像明显优于无序的 VQ 方案<sup>[60,61]</sup>。后一方案的改进程度不如前一种方案。

### 3.5.5 SOM 用于数据采掘和知识发现

这是最近几年来发展很快的领域 它受到 ANN、计算机科学（特别是数据库领域）统计学等许多学科的研究者高度重视并且通力合作。SOM 在其中起重要作用，由于篇幅限制本章不再作详细介绍，可参考绪论的 1.5 节及本章文献[70] [71]。

## 3.6 ART 的基本原理和算法实现框架

ART 的出发点是借鉴人的认知过程和大脑工作的特点。人可以在复杂、非平稳和有干扰的环境中学会对各种事物进行分类和识别。这种学习往往是自组织的。此外，人对于所学得知识的积累和存储既具有刚性又具有弹性，即一方面可以牢固地保存已学的知识，另一方面又能学大量新知识。人脑还具有集中注意力于“有用”或“重要”事物而忽略掉“无关”或“次要”事物的能力。人脑能够根据外界对于自身判断和分类的结果所作出的“奖”或“惩”回应来调整学习的策略。这一切表明，人脑的学习不简单地是外界  $\rightarrow$  内部的映射过程，这称为由底向上（bottom-up），而且还包括人的主观意志参与了学习过程，这称为由顶向下（top-down）。ART 作为一种模仿人脑认知过程的自组织聚类算法，在解决近代日益复杂的大量数据聚类 and 分类时，具有独特优点。其应用包括对于复杂系统的状态进行分类和判断，实现数据采掘（DM）和知识发现等。ART 的主要优点是聚类效果好而且稳定，此外还能高效利用系统的记忆资源。由于 ART 是由简单的“竞争学习机制”演化而来的，下面先介绍这种机制。

设有一个由  $M$  个神经元构成的网络，各神经元编号为  $j = 1 \sim M$ ，相应的输出记为  $y_j, j = 1 \sim M$ 。设网络的输入是  $N$  维列向量  $\mathbf{X} = [x_1, \dots, x_N]^T$ 。每个神经元  $j$  有一个相应的  $N$  维权向量  $\mathbf{W}_j, \mathbf{W}_j = [w_{j1}, \dots, w_{jN}]^T$ 。若按照节拍  $k = 1, 2, \dots$  向网络送入  $\mathbf{X}(k)$  那么可以采用由下列 3 步构成的竞争学习机制来调整各个权  $\mathbf{W}_j(k)$ ，以实现网络的聚类功能。

#### 1. 预处理

对任一节拍  $k, \mathbf{X}(k) = [x_1(k), \dots, x_N(k)]^T$  若满足条件  $0 \leq x_i(k) \leq 1, \forall i = 1 \sim N$  则无需处理。若此条件不满足 则应该用第 2 章所述线性变换使之满足此条件。

#### 2. 竞争

用下式计算节拍  $k$  的获胜端  $j^*(k)$ ：

$$j^*(k) = \operatorname{argmax}_{j=1 \sim M} [\mathbf{W}_j^T(k) \mathbf{X}(k)]$$

这时网络的输出为

$$y_j(k) = \begin{cases} 1, & j = j^*(k) \\ 0, & j \neq j^*(k) \end{cases}$$

这样可根据  $y_j(k)$  输出为 1 者判别  $\mathbf{X}(k)$  所属的类别 其标记为  $j^*(k)$ 。

### 3. 学习

按下式对诸权向量进行调整：

$$\mathbf{W}_j(k+1) = \begin{cases} \mathbf{W}_j(k) + \alpha[\mathbf{X}(k) - \mathbf{W}_j(k)], & j = j^*(k) \\ \mathbf{W}_j(k), & j \neq j^*(k) \end{cases}$$

其中  $\alpha$  是一个小正数（例如  $\alpha = 0.1$ ）。可以看到，只有竞争获胜者获得学习（即进行权调整）的机会，而其他神经元都无权学习。

竞争学习机制的重大缺陷是聚类性能不稳定。设有一个向量  $\mathbf{X}_a$  在一系列输入向量中前后两次出现（中间隔着若干其他向量），则有可能两次判别的类别归属标记不一致。这说明在这种简单的竞争学习机制中，旧的记忆内容有可能被新的记忆内容破坏。

为了克服上述缺陷，在简单竞争学习机制上增加了一个自稳机制，从而构成 ART。ART 有各种变型，下面先给出一个典型的 ART 系统的工作原理及算法框架，其他各种可能的变型将放在下一节讨论。

ART 算法由下列内容构成。

(1) 网络中的  $M$  个神经元分成两类：一类称为“已占用”另一类称为“未占用”。在学习开始以前所有神经元都是未占用的。

(2) 按节拍  $k = 1, 2, \dots$  依次送进网络的输入向量  $\mathbf{X}(k) = [x_1(k), \dots, x_N(k)]^T$ 。若  $0 \leq x_i(k) \leq 1, \forall i, k$ ，则无需进行任何处理。若不满足则应进行预处理，即用第 2 章介绍过的线性变换使其满足。

(3) 对于节拍  $k = 1$  实施下列操作：令

$$\mathbf{W}_1(k) = \mathbf{X}(k)$$

即赋值  $\mathbf{W}_1(1)$  为输入向量  $\mathbf{X}(1)$  之值且  $j = 1$  为已占用标号。

(4) 对于节拍  $k > 1$ ，实施竞争操作。设此时有  $\mu(k)$  个神经元已被占用，其标号为  $j = 1 \sim \mu(k)$ 。在这  $\mu(k)$  个神经元中进行竞争计算，找到第 1 获胜标号  $j_1^*(k)$  如下：

$$j_1^*(k) = \operatorname{argmax}_{j=1 \sim \mu(k)} [\mathbf{W}_j^T(k) \mathbf{X}(k)] \quad (3-57)$$

这一步称为由底向上操作。

(5) 比较  $\mathbf{W}_{j_1^*(k)}(k)$  和  $\mathbf{X}(k)$  的匹配程度 二者的匹配度  $\eta_1$  由下式计算：

$$\eta_1 = \frac{\mathbf{W}_{j_1^*(k)}^T(k) \mathbf{X}(k)}{\|\mathbf{W}_{j_1^*(k)}(k)\| \|\mathbf{X}(k)\|} \quad (3-58)$$

$\eta_1 \in [-1, 1]$ ,  $\eta_1 = 1$  为完全匹配  $\eta_1$  距 1 越远则匹配程度越差。为了判断二者的匹配度是否合格，设置一个警戒 (vigilance) 值  $\rho$ ,  $\rho$  是一个略小于 1 的数。若  $\eta < \rho$  表明匹配度不合格 转而进行下列 (6) 的操作；若  $\eta \geq \rho$ ，表明匹配度合格，这时称系统进入自适应谐振 (adaptive resonance) 状态，转而进行下列 (7) 的操作。上面所述的匹配过程称为由顶向下操作。

(6) 再匹配。由于  $\mathbf{W}_{j_1^*(k)}(k)$  与  $\mathbf{X}(k)$  的匹配度低于警戒值  $\rho$ ，所以  $j_1^*(k)$  被废除 再进行竞争以找到第 2 获胜标号  $j_2^*(k)$ ，

$$j_2^*(k) = \operatorname{argmax}_{j=1 \sim \mu(k), j \neq j_1^*(k)} [\mathbf{W}_j^T(k) \mathbf{X}(k)] \quad (3-59)$$

然后, 仿照第 5 步, 计算  $W_{j_2^*(k)}(k)$  与  $X(k)$  的匹配度  $\eta_2$  (参见式 3-58)。如果  $\eta_2 \geq \rho$  则转而进行第 (7) 步操作, 反之, 则找到第 3 获胜标号  $j_3^*(k)$  并计算  $W_{j_3^*(k)}(k)$  与  $X(k)$  的匹配度  $\eta_3$ 。依此类推, 若在各已占用的  $\mu(k)$  个神经元中终于找到 1 个匹配度合格的神经元, 则可继续按第 (7) 步进行操作。反之, 若一个也找不到, 那么首先检验:  $\mu(k) < M$  若此条件成立, 表明网络中尚有未占用端。这时可以将  $j = \mu(k) + 1$  从未占用标号改变成占用标号, 并且令  $W_{\mu(k)+1}(k) = X(k)$ 。若此条件不成立, 说明网络中所有神经元都被占用, 不可能再开辟一个新的标号来表示一种新的聚类。这种情况说明, 在规定的  $\rho$  值下网络记忆容量不足, 不能再接受新知识, 否则老的记忆内容将被破坏。为了对付这种情况, 可采取 3 种策略: 第 1 种, 节拍  $k$  的学习终止。第 2 种, 扩大记忆容量, 即加大  $M$ 。第 3 种, 减小  $\rho$  值。这种选择的后果将在下面讨论。

(7) 进入自适应谐振状态后进行权的学习。设  $j_l^*(k)$  为满足匹配条件的获胜端标号, 则可以按下式进行权向量的调整:

$$W_j(k+1) = \begin{cases} W_j(k) + \alpha[X(k) - W_j(k)], & j = j_l^*(k) \\ W_j(k), & j \neq j_l^*(k) \end{cases} \quad (3-60)$$

其中  $\alpha$  是一个小正数, 例如  $\alpha = 0.1$ 。注意, 与 SOM 的学习策略不同, 在该处  $\alpha$  是一个随  $k$  的增加而逐渐减小的步幅函数且最终趋向于 0, 因而不能适应于非平稳的环境。而 ART 的  $\alpha$  取为固定的步幅系数值, 因此可以适应于缓慢变化的非平稳环境。下面对于此算法中 (5)、(6) 两步由顶向下的匹配和再匹配的意义作出阐释并且讨论  $\rho$  的大小对于聚类效果的影响。

为什么在求得第 1 获胜端标号  $j_1^*(k)$  后还要作一次匹配运算呢? 为什么  $j_1^*(k)$  不满足匹配条件后, 第 2 获胜端  $j_2^*(k)$  甚至第 3 获胜端  $j_3^*(k)$  还能满足匹配条件呢? 其症结在于各个权向量的模值可能参差不齐。下面举例说明之。试比较一下图 3-10 中 (a) 和 (b) 两种情况。其中  $\|W_{j_1^*(k)}(k)\|$  大于  $\|W_{j_2^*(k)}(k)\|$ 。虽然  $X(k)$  与  $W_{j_2^*(k)}(k)$  的相似度高, 与  $W_{j_1^*(k)}(k)$  的相似度低, 但是  $W_{j_1^*(k)}^T(k)X(k)$  仍高于  $W_{j_2^*(k)}^T(k)X(k)$ 。这样  $j_1^*(k)$  成为第 1 获胜标号, 而  $j_2^*(k)$  反成为第 2 获胜标号。由于在计算匹配度  $\eta_1$  和  $\eta_2$  时 (参见式 (3-58)), 各权向量的模被归一化了, 所以反而有了  $\eta_2 > \eta_1$ 。这样很自然地会提出另一些问题: 第一, 能否将由底向上的竞争和由顶向下的匹配合而为一呢 (例如, 只采取匹配度作为惟一准则)? 第二, 能否采取其他竞争及匹配准则呢? 这将留在下一节关于各种变型 ART 的介绍中讨论。

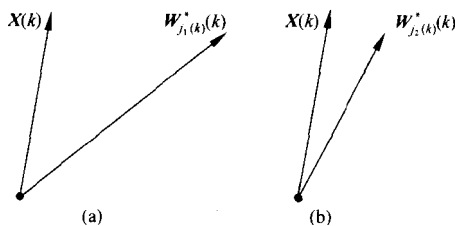


图 3-10 权向量的两种情况

在 ART 算法中, 警戒值  $\rho$  的选择十分重要。如果  $\rho$  选得很接近于 1 (例如  $\rho = 0.99$ ) 那么类别将分得非常细, 这时所需的网络记忆容量  $M$  将非常大。反之, 若  $\rho$  选得远小于 1 (例如  $\rho = 0.7$ ), 那么类别将分得较粗, 相应的  $M$  也较小。显然,  $\rho$  的选择取决于分类的精度与可用记忆容量  $M$  之间的协调。为此可取各种  $\rho$  的选择方案, 下面举其中二例。第一种方案, 先任选一个  $\rho$  值进行 ART 的学习。如果学习终结时, 尚有多余的记忆容量未用完, 这表明  $\rho$  值选得太小, 那么可略增加  $\rho$  值后再从头进行学习。此过程可继续下去直至记忆容量用完为止。如学习未终结时记忆容量已用完, 则说明  $\rho$  值太接近于 1 了, 这时应略

降低  $\rho$  值后再学习。此过程可继续到记忆容量够用为止。第二种方案，采取奖惩式的方法。首先任选一个  $\rho$  值，ART 按此值行学习和分类，并且将分类结果与 ART 所使用的环境进行双向交流。这就是 ART 将分类结果告知某个需要依据此结果进行决策或采取行动的主体。据此结果，该主体动作后，总是只有两种可能的后果——成功或失败。这时，该主体可向 ART 反馈回相应的奖或惩信息。奖信息表明分类过细，而惩信息表明分类过粗。ART 因而能及时调节  $\rho$  值来予以校正。这一过程实际上经常发生在人脑的认知和学习的经历中。例如，一个人可以先将任何颜色的香蕉划归于一种类别。当他吃了黄色和绿色香蕉并尝到后者涩不可食后，才发现二者迥然不同，这时才将它们划分为两种类别。

### 3.7 ART 的各种变型及其比较

现在常用的几种 ART 变型是 ART2A、ART2A-E、ART2A-C 和 Fuzzy ART 其中最后一种采用模糊算法的 ART 将放在后面第 5 章讨论 本节只讨论前面 3 种并对其特点和性能作出评价和比较。

#### 3.7.1 ART2A

这是 ART 发展过程中一种较完善而且便于实现的类型<sup>[62]</sup>，可以说代表了截至 1991 年为止的一项关于 ART 算法的总结性成果。它仍然由预处理、竞争-匹配和自适应学习这 3 部分组成。系统按节拍  $k$  运行， $k = 1, 2, \dots$ 。节拍  $k$  时，网络的输入为  $N$  维列向量  $\mathbf{X}(k) = [x_1(k), \dots, x_N(k)]^T$ 。网络含  $M$  个神经元，分成占用和未占用两类。设节拍  $k$  时，有  $\mu(k)$  个神经元被占用，相应这些神经元的权向量是  $\mathbf{W}_j(k) = [w_{j1}(k), \dots, w_{jN}(k)]^T$ ， $j = 1 \sim \mu(k)$  未占用的神经元未设定权向量。当开始运行即  $k = 1$  时  $\mu(1) = 0$  即所有神经元都是未占用的。下面讨论预处理、节拍  $k = 1$  及  $k > 1$  的竞争-匹配和自适应学习。

##### 1. 预处理

ART2A 的一项基本假设是，输入向量  $\mathbf{X}(k)$  的各分量  $x_i(k)$  永远满足下列要求：

$$0 \leq x_i(k) \leq 1 \quad i = 1 \sim N, \forall k \quad (3-61)$$

如果此要求不满足，则应采用第 2 章介绍过的线性变换使之得到满足。然后可以进行下列两种预处理中的任意一种。

##### (1) 归一化预处理

如果  $\|\mathbf{X}(k)\| = 1$  对任何  $k$  成立则无需采取措施。反之，则应该用归一化的输入向量  $\mathbf{X}(k)$  替代  $\mathbf{X}(k)$  ( $\|\mathbf{X}(k)\| \neq 1$ ) 即

$$\mathbf{X}(k) = \mathbf{X}(k) / \|\mathbf{X}(k)\| \quad (3-62)$$

##### (2) 对比增强-归一化处理

在 (1) 项归一化处理得到  $\mathbf{X}(k) = [\hat{x}_1(k), \dots, \hat{x}_N(k)]^T$  的基础上再作增强对比度的处理，即



$$f_i(k) = \begin{cases} \hat{x}_i(k), & \text{任何 } \hat{x}_i(k) > \theta \\ 0, & \text{其他} \end{cases} \quad (3-63)$$

其中  $\theta$  是一个小正数，其值的选择范围如下：

$$0 \leq \theta \leq 1/\sqrt{N} \quad (3-64)$$

这样可得到  $F(k) = [f_1(k), \dots, f_N(k)]^T$ 。这项措施达到两个目的 第一 当某个  $\hat{x}_i(k)$  小于  $\theta$  时，其值可与噪声和干扰值相比拟，其存在对于类别特征的辨识无益，故可去除。第二，对于聚类而言，起主要作用的是较大分量与较小分量之间的差距，去除微小分量使对比度得以增强。最后 再作一次归一化处理 用  $X(k)$  作为输入向量 即

$$\tilde{X}(k) = F(k) / \|F(k)\| \quad (3-65)$$

下面只以  $X(k)$  作为输入向量为例来进行讨论。

## 2. 节拍 $k=1$ 的竞争-匹配和自适应学习

这时没有一个神经元是被占用的，所以可简单地令  $W_1(1) = \hat{X}(1)$  且令  $\mu(2) = 1$ 。由于  $\|X(1)\| = 1$  所以  $\|W_1(1)\| = 1$ 。

## 3. 节拍 $k>1$ 的竞争-匹配和自适应学习

对于节拍  $k$  已占用神经元数为  $\mu(k)$  设其权为  $W_j(k), j = 1 \sim \mu(k)$ 。在下面即将给出的自适应学习算法可以保证  $\|W_j(k)\| = 1, \forall j, k$ 。这样 竞争和匹配度计算两项操作可以并成一步完成，通过下列计算求得的竞争第 1 获胜标号  $j_1^*(k)$  也就是匹配度最高者的标号，其匹配度为  $\eta_1$  计算公式如下：

$$\begin{cases} j_1^*(k) = \operatorname{argmax}_{j=1 \sim \mu(k)} [W_j^T(k) X(k)] \\ \eta_1 = W_{j_1^*(k)}^T(k) X(k) / \|W_{j_1^*(k)}\| \|X(k)\| \end{cases} \quad (3-66)$$

下面即可检验此匹配度是否达到预先设置的警戒值  $\rho$ 。若  $\eta_1 < \rho$  表明不合格 这时进一步检验记忆容量是否已被占满。若  $\mu(k) = M$ ，即已占满，这时不可能再新开辟一个新的聚类 此节拍  $k$  的学习不再进行并转入下一节拍。若  $\mu(k) < M$  即未占满 这时可开辟一个标号为  $\mu(k) + 1$  的新聚类区并且令  $W_{\mu(k)+1}(k) = X(k)$  注意其模为 1。若  $\eta_1 \geq \rho$  表明匹配度合格，这时进入自适应谐振状态，即可进行权向量的学习调整。与预处理部分相同，权的学习也可以分为归一化方式和对比增强-归一化方式两种，分别介绍如下。

第一种，按以下公式计算：

$$W_j(k+1) = \begin{cases} \tilde{W}_j(k+1) / \|\tilde{W}_j(k+1)\|, & \tilde{W}_j(k+1) = W_j(k) + \alpha[\hat{X}(k) - W_j(k)], \quad j = j_1^*(k) \\ W_j(k), & j \neq j_1^*(k) \end{cases} \quad (3-67)$$

可以看到，此学习算法能够保证：若  $\|W_j(k)\| = 1, \forall j = 1 \sim \mu(k)$ ，则必然有  $\|W_j(k+1)\| = 1, \forall j = 1 \sim \mu(k)$ 。

第二种，按以下公式计算：

$$\mathbf{W}_j(k+1) = \begin{cases} \textcircled{1} \boldsymbol{\Phi}_j(k+1) / \|\boldsymbol{\Phi}_j(k+1)\|, \\ \textcircled{2} \boldsymbol{\Phi}_j(k+1) = [\varphi_{j1}(k+1), \dots, \varphi_{jN}(k+1)]^T, \\ \textcircled{3} \varphi_{ji}(k+1) = \tilde{w}_{ji}(k+1), \quad \text{当 } \tilde{w}_{ji}(k+1) > \theta; \\ \textcircled{4} \varphi_{ji}(k+1) = 0, \quad \text{当 } \tilde{w}_{ji}(k+1) \leq \theta, i = 1 \sim N, \\ \textcircled{5} \tilde{\mathbf{W}}_j(k+1) = [\tilde{w}_{j1}(k+1), \dots, \tilde{w}_{jN}(k+1)] = \hat{\mathbf{W}}_j(k+1) / \|\hat{\mathbf{W}}_j(k+1)\|, \\ \textcircled{6} \hat{\mathbf{W}}_j(k+1) = \mathbf{W}_j(k) + \alpha[\hat{\mathbf{X}}(k) - \mathbf{W}_j(k)], \quad j = j_1^*(k); \\ \textcircled{7} \mathbf{W}_j(k), \quad j \neq j_1^*(k) \end{cases} \quad (3-68)$$

可以看到，这一公式包括了“权学习—归一化—对比增强—归一化”共4步计算，能保证  $\|\mathbf{W}_j(k+1)\| = 1, \forall j = 1 \sim \mu(k)$ 。式(3-67)和(3-68)中的步幅  $\alpha$  是一个固定的小正常数，例如  $\alpha = 0.1$ 。 $\theta$  仍可按式(3-64)所限定范围进行选择。

下面对 ART2A 算法做出评价。由于此算法中输入向量和各权向量都进行了归一化处理，模值皆等于1。这样，无需像3.6节所述的ART框架那样，将竞争计算和匹配度计算分开，从而避免了反复进行竞争和匹配，使计算效率大为提高。此外，此算法中还引入了增强对比度的选项。当环境噪声不能忽略时，采用此选项可以改善聚类效果。实际上，在此算法中进行竞争和匹配度比较的惟一准则是两个向量在  $N$  维空间中的夹角（因为无论输入向量还是权向量都已归一化为具有单位模值的向量，只有角度成为其惟一特征）。这种做法虽然可以大大简化计算，但是丢失了幅度信息。如果幅度信息不允许丢失，则此种算法不可取。而下面介绍的 ART2A-E 和 ART2A-C 都可以克服此缺点。

### 3.7.2 ART2A-E

此算法的前提与 ART2A 完全相同，下面也分为3项讨论其计算过程。

#### 1. 预处理

ART2A-E 对于输入向量  $\mathbf{X}(k)$  的要求是只需使其满足式(3-61)，而不作其他任何处理。

#### 2. 节拍 $k=1$ 的竞争-匹配和自适应学习

与 ART2A 相同，只需令  $\mathbf{W}_1(1) = \mathbf{X}(1)$  且令  $\mu(2) = 1$ 。注意，由于  $\mathbf{X}(k)$  未作归一化处理， $\|\mathbf{W}_1(1)\|$  不一定等于1。

#### 3. 节拍 $k>1$ 的竞争-匹配和自适应学习

竞争和匹配度计算并成一步完成，用下列计算求得第1获胜标号  $j_1^*(k)$ ，这也就是匹配度最高者的标号，其匹配度为  $\eta_1$ 。计算公式为

$$\left. \begin{aligned} j_1^*(k) &= \underset{j=1 \sim \mu(k)}{\operatorname{argmin}} [\|\mathbf{X}(k) - \mathbf{W}_j(k)\|] \\ \eta_1 &= 1 - \frac{\|\mathbf{X}(k) - \mathbf{W}_{j_1^*(k)}(k)\|}{\sqrt{N}} \end{aligned} \right\} \quad (3-69)$$

注意， $\|\cdot\|$  表示求模， $\|\mathbf{X}(k) - \mathbf{W}_{j_1^*(k)}(k)\|$  表示  $\mathbf{X}(k)$  与  $\mathbf{W}_{j_1^*(k)}(k)$  的欧氏距离，它可用下式计算：

$$\| \mathbf{X}(k) - \mathbf{W}_{j^*(k)}(k) \| = \left[ \sum_{i=1}^N (x_i(k) - w_{j^*(k)i}(k))^2 \right]^{\frac{1}{2}}$$

可以看到, 若  $\mathbf{X}(k) = \mathbf{W}_{j^*(k)}(k)$ , 则  $\eta_i = 1$ , 若  $\mathbf{X}(k)$  与  $\mathbf{W}_{j^*(k)}(k)$  正交且模值皆为 1, 则  $\eta_i = 0$ .

下面比较  $\eta_i$  与  $\rho$ 。若  $\eta_i < \rho$ , 则应开辟一个新的聚类或停止本拍学习, 方法与 ART2A 相同; 若  $\eta_i \geq \rho$ , 则进入自适应谐振状态, 这时可按下式进行权向量的调整:

$$\mathbf{W}_j(k+1) = \begin{cases} \mathbf{W}_j(k) + \alpha [\mathbf{X}(k) - \mathbf{W}_j(k)], & j = j_i^*(k) \\ \mathbf{W}_j(k), & j \neq j_i^*(k) \end{cases} \quad (3-70)$$

可以看到, 各  $\mathbf{W}_j(k+1)$  的模值不一定等于 1。

下面对此算法作出评价。由于此算法对于输入向量和各权向量都不作归一化处理, 因此幅度信息仍然保留。在匹配度的计算中比较的是两个向量的欧氏距离, 这就是说只有当二者的夹角和二者的幅度差都比较小时, 其匹配度才可能比较高。对于幅度信息不可忽略的情况, 这是一种合理的选择。此外, 此算法还具有计算过程简单和计算量小的优点。

### 3.7.3 ART2A-C

此算法的前提仍与 ART2A 相同。为了解决不丢失幅度信息的问题, 采取了扩充补码向量的方案。下面仍分 3 部分介绍这一算法。

#### 1. 预处理

仍按照式 (3-61) 要求  $\mathbf{X}(k)$  的每一分量  $x_i(k) \in [0, 1]$ 。其次形成  $\mathbf{X}(k)$  的补码向量  $\mathbf{X}^C(k) = [x_1^C(k), \dots, x_N^C(k)]^T$  其中  $x_i^C(k) = 1 - x_i(k), i = 1 \sim N$ 。然后将  $\mathbf{X}(k)$  和  $\mathbf{X}^C(k)$  并成一个  $2N$  维列向量  $\hat{\mathbf{X}}(k), \hat{\mathbf{X}}(k) = [\mathbf{X}^T(k), \{\mathbf{X}^C(k)\}^T]^T$ 。试举一例: 设有两个 2 维输入向量  $\mathbf{X}_a = [0.1, 0.3]^T$  和  $\mathbf{X}_b = [0.3, 0.9]^T$  它们的夹角为 0 而模值不同。相应的扩展向量为  $\hat{\mathbf{X}}_a = [0.1, 0.3, 0.9, 0.7]^T$  和  $\hat{\mathbf{X}}_b = [0.3, 0.9, 0.7, 0.1]^T$ 。可以看到, 它们之间的夹角既反映了  $\mathbf{X}_a$  和  $\mathbf{X}_b$  之间的角度差异又反映了幅度差异。现在即以  $\hat{\mathbf{X}}(k)$  作为网络输入。

#### 2. 节拍 $k=1$ 的竞争-匹配和自适应学习

由于输入向量已扩展为  $2N$  维, 网络各神经元的权向量  $\mathbf{W}_j(k)$  也应是  $2N$  维的。当  $k=1$  时, 只需令  $\mathbf{W}_1(1) = \hat{\mathbf{X}}(1)$  且令  $\mu(2) = 1$ 。

#### 3. 节拍 $k>1$ 的竞争-匹配和自适应学习

竞争和匹配度计算并成一步完成, 其计算公式如下:

$$\left. \begin{aligned} j_i^*(k) &= \operatorname{argmax}_{j=1 \sim \mu(k)} \left[ \frac{\mathbf{W}_j^T(k) \hat{\mathbf{X}}(k)}{\|\mathbf{W}_j(k)\| \|\hat{\mathbf{X}}(k)\|} \right] \\ \eta_i &= \frac{\mathbf{W}_{j_i^*}^T(k) \hat{\mathbf{X}}(k)}{\|\mathbf{W}_{j_i^*}(k)\| \|\hat{\mathbf{X}}(k)\|} \end{aligned} \right\} \quad (3-71)$$

若  $\eta_i < \rho$ , 则开辟一个新聚类或停止本拍学习 (与 ART2A 同); 若  $\eta_i \geq \rho$  则进入自适应谐振状态并按照下式进行权向量的调整:

$$\mathbf{W}_j(k+1) = \begin{cases} \mathbf{W}_j(k) + \alpha [\hat{\mathbf{X}}(k) - \mathbf{W}_j(k)], & j = j_1^*(k) \\ \mathbf{W}_j(k), & j \neq j_1^*(k) \end{cases} \quad (3-72)$$

易于证明（见文献[63]、[64]）如果各  $\mathbf{W}_j(k)$  是由  $N$  维列向量  $\tilde{\mathbf{W}}_j(k)$  及其补码向量  $\tilde{\mathbf{W}}_j^c(k)$  构成的，即  $\mathbf{W}_j(k) = [\tilde{\mathbf{W}}_j^T(k), \{\tilde{\mathbf{W}}_j^c(k)\}^T]^T$  那么各  $\mathbf{W}_j(k+1)$  也是由相应的  $N$  维列向量  $\mathbf{W}_j(k+1)$  及其补码向量  $\tilde{\mathbf{W}}_j^c(k+1)$  构成的。这样对于任何  $k$  值，各权向量  $\mathbf{W}_j(k)$  都是由  $\tilde{\mathbf{W}}_j(k)$  和  $\tilde{\mathbf{W}}_j^c(k)$  两部分构成的。

可以看到，按照此算法形成的各聚类区的权向量实际是区内各个按增加补码部分扩展为  $2N$  维的输入向量的质心。当一个  $2N$  维扩展输入向量与某个权向量的夹角最小时，即划归于相应的聚类区。

### 3.7.4 各种 ART 算法的比较

文献[64]给出了关于各类 ART 算法聚类性能的两项模拟实验结果（包括 Fuzzy ART）。下面简要介绍它们并据此对各种算法的性能作出比较。

第一项实验的输入是 2 维列向量  $\mathbf{X} = [x_1, x_2]^T$ ，训练集中包含 100 个样本，其中  $x_1$  和  $x_2$  各取 10 个离散值：0.1, 0.2, ..., 0.9, 1.0。在网络进行学习时，将训练集中的样本随机组合成两种序列，称为 A 和 B，每种序列中每个样本至少出现 10 次。学习采用固定步幅  $\alpha = 0.1$ ，警戒值  $\rho$  取不同数值且最大记忆容量不受限。图 3-11 显示了 ART2A、ART2A-C、ART2A-E 这 3 种算法的模拟实验结果。图中的小圆表示训练样本， $\times$  符号表示各权向量，各深色区表示聚类区。需要说明的是 ART2A-C 中的输入向量和权向量都已扩展为 4 维的，但是其另一半是前一半的补码，所以只需像图中那样画出前一半的 2 维向量即可。实验中所取的步幅都是  $\alpha = 0.1$ 。

从图 3-11(a) ~ (c) 可以看到，ART2A 算法是按照输入向量的角度不同实现聚类的，幅度信息则完全不起作用。还可以看到，当  $\rho$  值设为较低的 0.94 时，只形成了 3 个聚类区（即只有 3 个神经元被占用），如 (a) 所示。当  $\rho$  值设为较高的 0.98 时，形成 5 个聚类区（对于序列 A），如 (b) 所示，或形成 6 个聚类区（对于序列 B），如 (c) 所示。从图 3-11(d) ~ (i) 可以看到，ART2A-C 和 ART2A-E 两种算法形成的聚类区的形状比较相似，即按照各个输入向量样本之间的欧氏距离大小进行分割。由这些图也可以看到，随着  $\rho$  的增大，聚类区数（即被占用神经元数）相应增大。对于相同的  $\rho$  值，不同的训练样本序列所形成的聚类区的个数和形状也略有差异。此外，由 (g) ~ (i) 可以看到，对于 ART2A-E 算法，当  $\rho$  取一定值时，虽然不同训练序列形成的聚类区不同，但是任一权向量与其最邻近权向量之间的欧氏距离大致相同。当  $\rho$  值增大时，此欧氏距离缩小。

第二项实验中输入向量  $\mathbf{X}$  的维数  $N = 100$ ，它取自一个如下式描述的阶跃响应函数  $f(t)$  的前 100 个取样值（即  $f(0), f(1), \dots, f(99)$ ）：

$$f(t) = 0.5 \left\{ 1 - e^{-\zeta \cdot 2\pi f_0 t} \left[ \cos(2\pi f_0 t \sqrt{1 - \zeta^2}) + \frac{\zeta}{\sqrt{1 - \zeta^2}} \sin(2\pi f_0 t \sqrt{1 - \zeta^2}) \right] \right\}$$

$$\mathbf{X} = [x_1, x_2, \dots, x_{100}]^T, \quad x_i = f(t) |_{t=i} \in [0, 1], \quad i = 1 \sim 100$$

在实验中衰减系数  $\zeta$  和周期  $1/f_0$  各取 10 种不同数值，从而形成 100 种不同的 100 维输入

向量构成的样本集。其中  $\zeta = 0.1, 0.13, 0.16, 0.21, 0.27, 0.34, 0.43, 0.55, 0.71, 0.9$  ,而  $1/f_0 = 10, 20, 30, \dots, 100$ 。实验中也分 A、B 两个样本序列，每个序列中各样本随机出现，但任何一个样本出现的次数不少于 20 次。学习的步幅都是  $\alpha = 0.1$ 。图 3-12 给出了 ART2A（图中 (a) 和 (b)）以及 ART2A-E（图中 (c)~(f)）两种算法的实验结果（聚类区是以  $\zeta$  和  $1/f_0$  来描述的，而不是原始的 100 维输入向量），非常有趣的是 对于此高维输入向量，这两种算法形成的聚类区形状的差异并不大（因为幅度不是敏感特征）。

文献[64] 还给出了 Fuzzy ART 算法用于这两项实验的模拟结果，其效果不如各种 ART2A 算法。

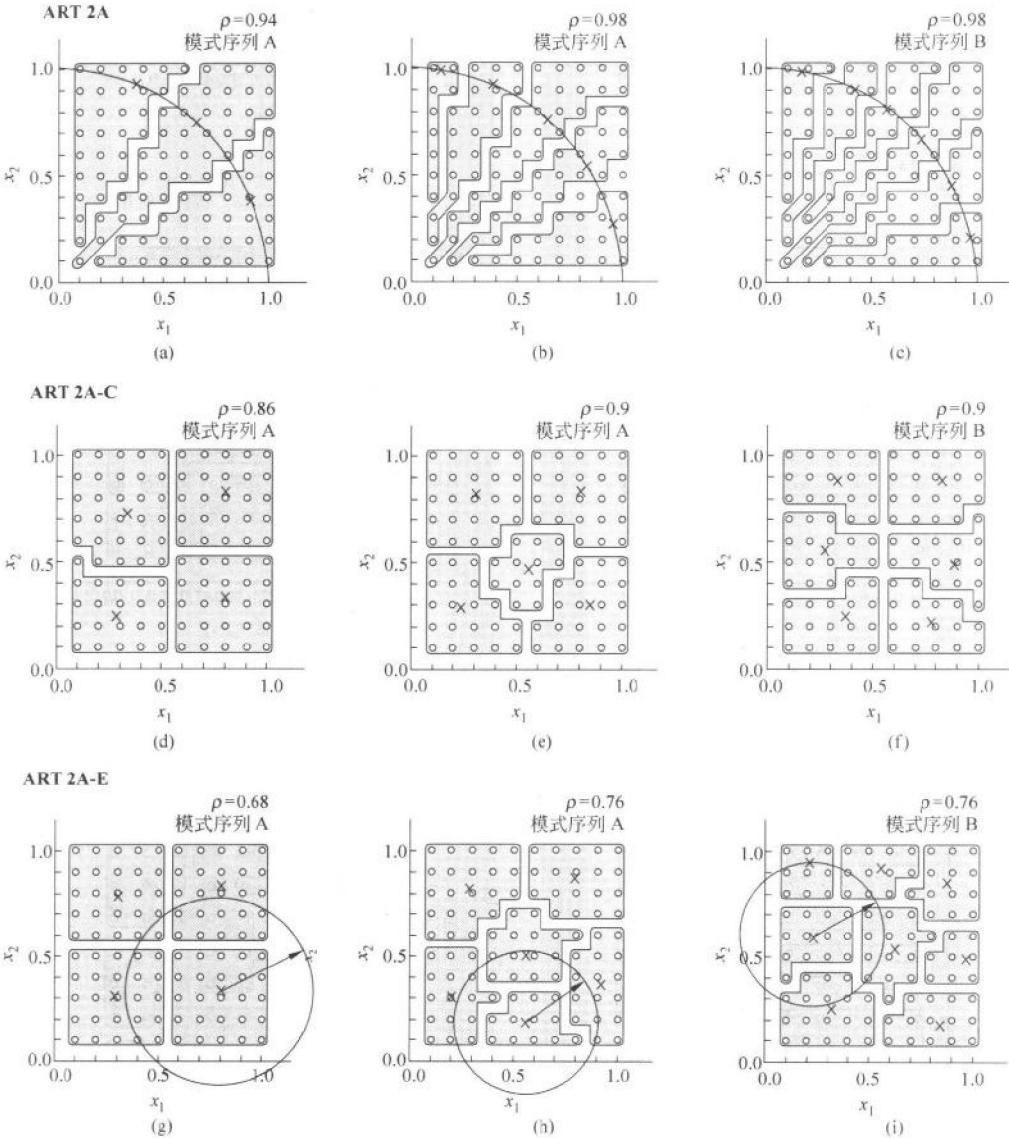


图 3-11 ART2A、ART2A-C、ART2A-E 算法的模拟实验结果

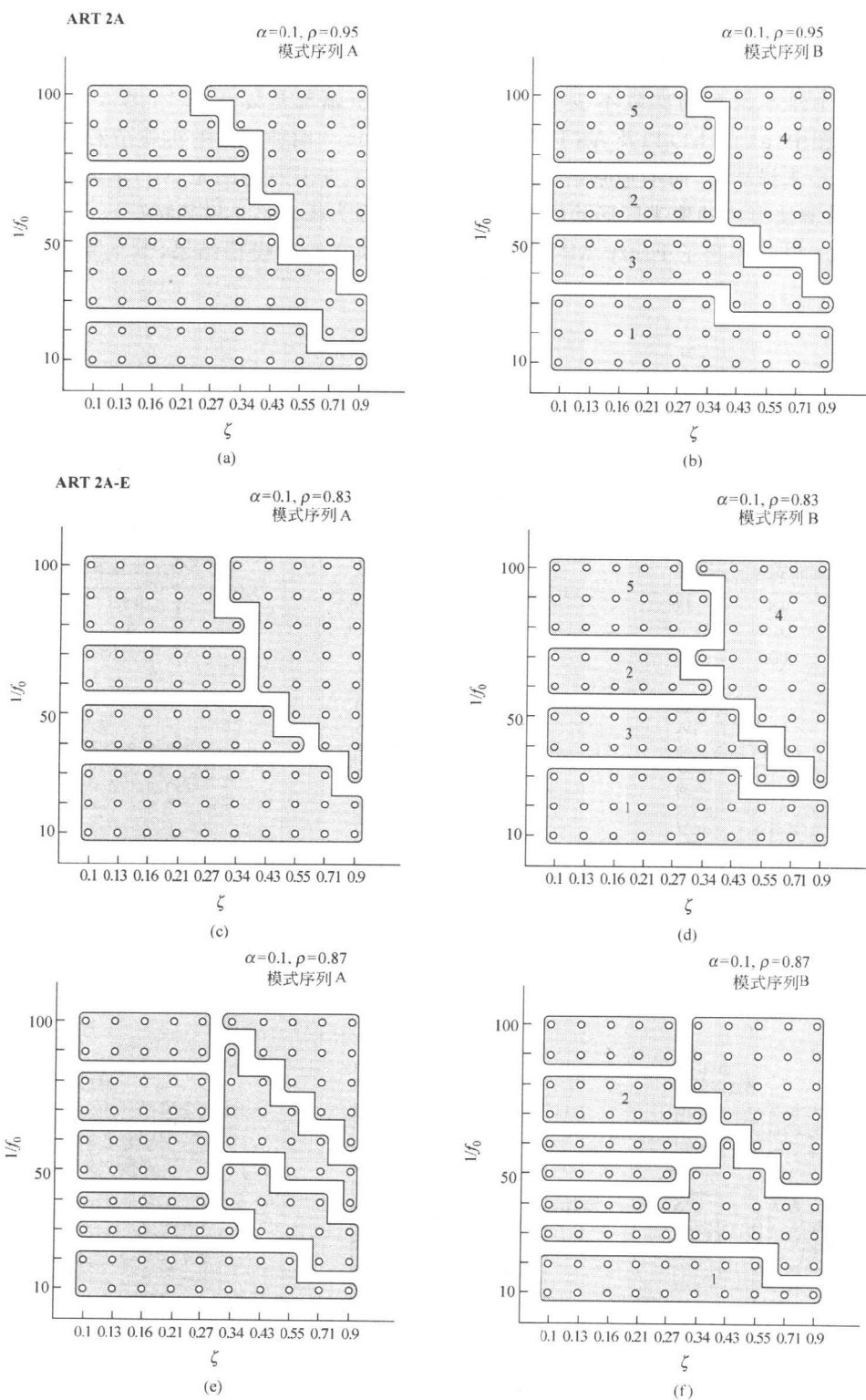


图 3-12 ART2A 和 ART2A-E 算法的实验结果

### 3.8 ART 应用举例 —— 在化学工业中的应用

对于一个复杂的过程性工业系统（例如，化工企业、核反应堆、医药工业、飞行中的飞机……）都需要通过若干传感器来监视系统运行的状况并且通过这些信号判断系统的运行是否正常。无论是由人还是由计算机（智能控制器）来操纵和控制这些系统时，都需要做到：故障检测（判断有无故障）、故障诊断（如发生了故障时判断故障的部位）、故障避免（如有故障的预兆而尚未发生时应及时采取措施予以避免）以及制造优化（使产品质量合格）。对于一个人而言，作为操作员他首要考虑的是辨识和避免任何（由人为因素、设备因素和环境因素引起的）潜在危险情况，任何危及安全的错误都是绝对不能容忍的，他必须百分之百地保证系统处于正常状态。如果采用机器智能操作器来代替人来进行操作，当然也应该做到这一点，但这非常困难，成了智能控制系统的一个瓶颈。由于大部分工业系统的运行表现出较慢的动力特性——其时间尺度是以分、时来计算的，这就有可能通过一个滑动时间窗口来取出各个传感器信号在窗口内的轨迹，从而判断该系统在运行过程中的行为。例如窗口宽度为 5.1 分钟，每 0.3 分钟取一个观察样点，这样每个传感器信号在窗口内取 17 个样点。若有 3 个传感器，那么共可取得 51 个信号值，它们即构成了判断滑动窗口所处时刻的系统行为的依据。这样，即可构成一个 51 维列向量作为智能判断系统的输入向量  $\mathbf{x}$ 。系统的行为主要是“正常”（无故障且产品质量合格）和“不正常”（有故障或产品质量不合格）。但是这两种状态不是截然分开的，从前者过渡到后者要经过一个“过渡”状态。此外，在系统运行过程中总可能发生一些在过去的经验中从未遇到过的情况，这属于“未定”状态（default）。一个智能判断系统就是要根据各传感器提供的输入向量  $\mathbf{x}$  来给出上面 4 种可能状态中的一种。当然，在得到了“正常”状态以外的其他结果时就应采取其他措施，例如判断故障的出处以及应该采用何种纠正措施等等，此外不再赘述。

实现智能判断需完成两项工作。第一项，将  $\mathbf{x}$  的赋值空间分成很多聚类区；第二项，将每个聚类区予以编号并贴上某一种状态的标签，这样即可根据某个  $\mathbf{x}$  所属聚类区的编号来判断系统的状态。其中第一项工作是关键，为完成此任务可采用本章介绍过的 SOM 网络，第 2 章介绍过的 RBF 网络以及第 5 章将介绍的模糊聚类网络（如 Fuzzy ART）等等。而这一节将介绍采用 ART2A-C 的一种方案（ART2A-E 等也都适用）。这种方案的独特优势是，聚类区的大小可以非常灵活地自由控制，对于环境的变化可以良好地自适应，而且算法非常简单高效，因此，已在很多过程状态辨识中得到应用。例如，飞行诊断<sup>[65]</sup>、核反应堆监视<sup>[66]</sup>以及合成孔径雷达系统图像中的目标辨识<sup>[67]</sup>等。下面将介绍的是在化学工业中应用的例子<sup>[63]</sup>

这里讨论的是一个化学工业中的模拟循环反应堆系统。为了去除化学反应中产生的热量，采用了一个外部再循环回路，通过外部注入冷却剂的循环来实现热交换。系统通过 3 个传感器提供的参数来监控其运行状态，它们是：送料速度、送料温度和冷却剂流速。反应堆的正常状态是指运行正常且产品的质量合格。一般情况下，送料速度应在  $0.36\text{m}^3/\text{min} \sim 0.4\text{m}^3/\text{min}$  范围内，送料温度应在  $64^\circ\text{C} \sim 66^\circ\text{C}$  范围内。控制系统可以在一个较宽范围内调节冷却剂的流速来保持系统处于正常状态。在实际运行过程中，由于逆流

(upstream) 处理问题，送料温度有可能突然增加 5 并且伴随着测量不到的送料纯度发生了变化。反应堆的控制器能够通过调节冷却剂的流速来补偿这种送料温度的突变。但是，如果控制失当，就会造成送料温度和冷却剂流速搭配不适合的组合干扰，从而引起产品质量的不合格。有经验的操作人员通过监视送料温度突变时冷却剂流速变化的特点来判断系统状态是否正常。现在，可以用 ART2A-C 来实现这个任务。首先，取一个宽度为 8.1 分钟的滑动窗口，在窗口内对 3 个传感器信号每隔 0.3 分钟采样一次，共得到 81 个采样值，它们构成了 ART2A-C 的 81 维输入列向量  $\mathbf{X} = [x_1, x_2, \dots, x_{81}]^T$ 。其次，将这个向量通过增加补码扩展为一个 162 维列向量  $\hat{\mathbf{X}} = [\mathbf{X}^T, \{\mathbf{X}^c\}^T]^T$  这是 ART2A-C 实际操作的输入向量。为了区分“正常”、“不正常”和“不确定”3 种状态，做了若干模拟实验。在一个实验中训练集由 3750 个  $\mathbf{X}$  的样本构成，在学习时从训练集中随机地取出各个样本（每个样本只取一次）构成一个序列依次送入 ART。警戒值设置为  $\rho = 0.99996$ 。经过 5 轮学习（一轮为整个序列送入一次）后共形成了 303 个聚类区，其中“正常”、“不确定”和“不正常”的聚类区数分别是 105、131 和 67<sup>[68]</sup>。在 5 轮学习后，所有训练集样本中只有 0.5% 的样本出现了既属于一个聚类区又属于另一个聚类区的情况（这就是说这种样本与两个不同聚类区的匹配度  $\eta$  值都超过了  $\rho$ ）但是这些不同的聚类区都属于同一类别（例如“正常”），这一现象表明有少数聚类区略有交迭。

在网络学习完成后进入工作时，用训练集以外的样本来进行测试。工作的准则是：当某个测试样本落入一个聚类区时，即赋予该聚类区在学习时确定的类别（如“不正常”）；当某个样本不属于任何聚类区时，系统给出结论为“不知道”，但是可以给出与其相似度最高的某个聚类区的类别。在所有测试中都不曾发生将“正常”与“不正常”混淆的情况。此系统还有能力进一步模仿人类专家通过自适应调整来适应环境的缓慢变化<sup>[69]</sup>



## 参 考 文 献

- [1] Kohonen T. Self-organization and associative memory. 3rd ed. Berlin-Heidelberg: Springer, 1989
- [2] Kohonen T, et al. Engineering applications of the self-organizing map. *PIEEE*, Oct. 1996, 84(10) : 1358 ~ 1384
- [3] 杨行峻, 郑君里. 人工神经网络. 北京: 高等教育出版社, 1992
- [4] Carpenter G A, Grossberg S. ART2: self-organization of stable category recognition codes for analog input patterns. *Appl. Opt.* 1987. 26: 4919 ~ 4930
- [5] Carpenter G A, Grossberg S. The ART of adaptive pattern recognition by self-organizing neural network. *Trans. IEEE on Computer*, March, 1988. 77 ~ 88
- [6] Carpenter G A, et al. ARTMAP: supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks*, 1991, 4: 565 ~ 588
- [7] Carpenter G A, et al. ART2-A: an adaptive resonance algorithm for rapid category learning and recognition. *Neural Networks*, 1991. 4: 493 ~ 504
- [8] Frank T, et al. Comparative Analysis of fuzzy ART and ART-2A network clustering performance. *Trans. IEEE on Neural Networks*, May, 1998, 9(3): 544 ~ 559
- [9] Carpenter G A, et al. Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 1991. 4: 759 ~ 771
- [10] Lin Simin, Sie Jennie. Weight-value convergence of the SOM algorithm for discrete input. *Neural Computation*, 1998, 10: 807 ~ 814
- [11] Haese Karin. Self-organizing feature maps with self-adjusting learning parameters. *IEEE Trans. on Neural Networks*, Nov. 1998. 9(6): 1270 ~ 1278
- [12] 杨行峻, 迟惠生等. 语音信号数字处理. 第2版. 北京: 电子工业出版社, 1998
- [13] Kohonen T. Self-organizing maps; optimization approaches. In: Kohonen T, et al, ed. *Artificial neural networks*. Amsterdam: North-Holland, 1991. 1: 981 ~ 990
- [14] Erwin E, et al. Self-organizing maps; ordering, convergence properties and energy functions. *Biol. Cybern.* 1992. 67(1): 35 ~ 45
- [15] Bouton C, Page's G. Self-organization and a. s. Convergence of the one-dimensional Kohonen algorithm with non-uniformly distributed stimuli. *Stochastic Process Appl*, 1993. 47: 249 ~ 274
- [16] Horowitz R, Alvarez L. Convergence properties of self-organizing neural networks. *Proc. of the 1995 American Control Conf.* 1995. 2: 1334 ~ 1339
- [17] Lihovidov V. Variational approach to unsupervised learning algorithm of neural networks. *Neural Networks*, 1997. 10(2): 273 ~ 289
- [18] Li X, et al. On the topology distortion in self-organizing feature maps. *Biol. Cybern.* 1993. 70: 189 ~ 198
- [19] Ruck D W, et al. Comparative analysis of backpropagation and the extended Kalman filter for training multilayer perceptrons. *Trans. IEEE on PAMI*, 1992. 10(4): 686 ~ 691
- [20] Maybeck P S. Stochastic models estimation and control. New York: Academic, 1979. 1: 1982. 2
- [21] Bauer H-U, Pawelzik K R. Quantifying the neighborhood preservation of self-organizing feature maps. *IEEE Trans. Neural Networks*, 1992. 3: 570 ~ 579

- [22] Villmann T, et al. Topology preservation in self-organizing feature maps: Exact definition and measurement. *IEEE Trans. Neural Networks*, 1997, 8; 256 ~ 266
- [23] Bezdek J C, Pal N R. An index of topological preservation for feature extraction. *Pattern Recognition*, 1995, 28(3): 381 ~ 391
- [24] Kohonen T. Self-organizing maps. Berlin: Springer-Verlag, 1994
- [25] Martinetz T, Schulten K. Topological representing network. *Neural Networks*, 1994, 7(3): 507 ~ 522
- [26] Herrmann M, et al. Measuring topology preservation in maps of real-world data. In: *European Symp. Artificial Neural Networks (ESANN'97)*. Bruges, 1995. 205 ~ 210
- [27] Herrmann, M, et al. A comparison of topology measures for neural maps. In: *Proc. Wkshp. Self-organizing Maps (WSOM'97)*. 1997. 274 ~ 279
- [28] Bauer H-U, et al. Controlling the magnification factor of self-organizing feature maps. *Neural Computation*, 1996, 8; 757 ~ 765
- [29] Wu Lizhong, Fallside Frank. On the design of connectionist vector quantizers. *Computer Speech and Language*, 1991, (6): 207 ~ 229
- [30] Kohonen T. Learning vector quantization. *Neural Networks*, 1988, 1(1): 303
- [31] Kohonen T, et al. Statistical pattern recognition: benchmarking studies. In: *Proc. of the IEEE International Conference on Neural Networks*. San Diego; 1988. 516 ~ 523
- [32] Renesh Padma, et al. A New connected word recognition algorithm based on HMM/LVQ segmentation and LVQ classification. In: *Proc. of IC'ASSP*, 1991, S2.19: 113 ~ 116
- [33] Monostori L, Bothe A. Convergence behavior of connectionist model in large scale diagnostic problems. In: *5th Int. Conf. on Industrial and Engineering Applications of Artif. Intell. and Expert Syst.* Berlin- Hiderberg; Springer, 1992, 113 ~ 122
- [34] Skitt P J C, et al. Process monitoring using auto-associative feedforward artificial neural networks. *J. Intell. Manuf.*, 1993, 4(1): 79 ~ 94
- [35] Firenze F, et al. Self-organizing networks; A challenging approach to fault diagnosis of industry processes. In: *Proc. ICANN'94*. London: Springer, 1994, 2, 1239 ~ 1242
- [36] Sorsa T, Koivo H N . Application of Artificial neural networks in process fault diagnosis. *Automatica*, 1993, 29(4): 843 ~ 849
- [37] Furukawa H, et al. A systematic method for rational definition of plant diagnostic symptoms by self-organizing neural networks. In: *Proc. Int. Conf. on Fuzzy Logic, Neur, Nets. and Soft computing*. Iizuka, Japan, Fuzzy Logic Syst. Inst. 1994. 555 ~ 556
- [38] Wu J-M, et al. Diagnosis for machine vibrations based on self-organizing neural network. In: *Proc. IECON'91, Int. Conf. on Industrial Electron., Contr. and Instrumentation*. Piscataway, NJ. 1991. 2, 1506 ~ 1510
- [39] Ballard D, Brown C. *Computer Vision*. Englewood Cliffs, NJ; Prentice-Hall, 1982
- [40] Lampinen J, Oja E. Distortion tolerant pattern recognition based on self-organized feature extraction. *IEEE Trans. on Neural Networks*, 1995, 6; 539 ~ 547
- [41] Oja E, Lampinen J. Unsupervised learning for feature extraction. In: Zurada J M, et al, ed *Computational Intelligence: Imitating Life*. New York; IEEE Press, 1994. 13 ~ 22
- [42] Buhmann J, et al. Distortion invariant object recognition by matching hierachically labeled graphs. In: *Proc. IJCNN'89 Piscataway (NJ)*; 1989. 1, 155 ~ 159
- [43] Visa A. Identification of stochastic textures with multiresolution features and self-organizing maps.

- In: Proc. 10th Int. Conf. on Pattern Recognition. Atlantic City, NJ; June, 1990. 518 ~ 522
- [44] Visa A. A texture classifier based on neural network principles. In: Proc. Int. Joint Conf. on Neural Networks. San Diego, CA; June, 1990. 1, 491 ~ 496
- [45] Kohonen T, et al. Statistical pattern recognition with neural networks; Benchmarking studies. In: Proc. IEEE Int. Conf. on Neur. Networks. San Diego (CA) July, 1988. 1, 61 ~ 68
- [46] Kohonen T. Improved version of learning vector quantization. In: Proc. IEEE Int Conf. on Neur. Networks. San Diego, CA; June 1990. 1, 545 ~ 550
- [47] Connors R W, Harlow C A. Equal probability quantizing and texture analysis of radiographic images. Computer Graphics and Image Process. Dec. , 1978, 8(3): 447 ~ 463
- [48] Haralick R M. Statistical and stracture approaches to texture. In: Proc IEEE. May 1979. PROC-67, 786 ~ 804
- [49] Haralick R M. Statistical image texture analysis. In: Young T Y, Fu K-S , ed Handbook of Pattern Recognition and Image Processing. New York; Academic, 1986
- [50] Gool L V, et al. Survey texture analysis anno 1983. Computer Vision, Gaphics and Image Processing, 1985, 29: 336 ~ 357
- [51] Connors R W, Harlow C A. A theoretical comparison of texture algorithms. IEEE Trans on PAMI, May, 1980, 2: 204 ~ 222
- [52] Haralick R, et al. Textural features for image classification. IEEE Trans on SMC, Nov. 1973, 3: 610 ~ 621
- [53] Weszka J S, et al. A comparative study of texture measures for terrain classification. IEEE Trans. on SMC, Apr 1976, 6: 269 ~ 285
- [54] Kohonen T. The self-organizing map. Proc. IEEE, 1990, 78: 1464 ~ 1480
- [55] Kohonen T. The “neural” phonetic typewriter. Computer, 1988, 21(3): 11 ~ 22
- [56] Mäntysalo J, et al. Mapping context dependent acoustic information into context independent form by LVQ. Speech Commun. , 1994, 14(2): 119 ~ 130
- [57] Corral J A, et al. Image compression via optimal vector quantization; A comparison between SOM, LBG and k-means algorithms. In: Proc. ICNN-94, Int. Conf. on Neur. Networks. Piscataway, NJ; IEEE 1994 6, 4113 ~ 4118
- [58] Nasrabadi N M, King R A Image coding using vector quantization; A review. IEEE Trans. on Comm. , 1988 36(8): 957 ~ 971
- [59] Nasrabadi N M, Feng Y. Vector quantization of images based upon the Kohonen self-organization feature maps. Neural Networks, 1988, 1(1): 518
- [60] Kangas J, Kohonen T. Developments and applications of the self-organizing map and related algorithms. In: Proc. IMACS Int. Symp. on Signal Processing, Robotics and Neur. Networks. Lille, France; 1994. 19 ~ 22
- [61] Kangas J. Increasing the error tolerance in transmission of vector quantized images by self-organizing map. In: Proc. ICANN'95, Int. Conf. on Artificial Neural Networks, Paris, France; 1995. 1287 ~ 291
- [62] Carpenter G A, et al. ART2-A: an adaptive resonance algorithm for rapid category learning and recognition. Neural Networks, 1991, 4: 493 ~ 504
- [63] Whitely J R, et al. Observations and problems applying ART2 for dynamic sensor pattern interpretation. IEEE Trans. Syst. , Man, Cybern. , 1996, 26: 423 ~ 437

- [64] Frank T, et al. Comparative analysis of fuzzy ART and ART-2A network clustering performance. IEEE Trans. on Neural Networks, 1998, 9(3): 544 ~ 559
- [65] Mc Duff R J, Simpson P K. An adaptive resonance diagnostics system. J. Neural Network Comp. 1990, 2: 19 ~ 29
- [66] Keyvan S, et al. Nuclear reactor condition monitoring by adaptive resonance theory. In: Proc. Int. Joint Conf. on Neural Networks. Baltimore, MD; 1992. I, 725 ~ 730
- [67] Fogler R J, et al. Feature discovery via neural networks for object recognition in SAR imagery. In: Proc. Int. Joint Conf. on Neural Networks. Baltimore MD; 1992. IV, 408 ~ 413
- [68] Whiteley J R, Davis J F. Qualitative interpretation of sensor patterns. IEEE Expert, 1993, 8: 54 ~ 63
- [69] Whiteley J R, Knowledge-based interpretation of process sensor patterns:[ph. D. Dissertation] . Columbus: Dept. of Chemical Engineering, The Ohio State Univ. , 1991
- [70] Special issue on neural networks for data mining and knowledge discovery. IEEE Trans. on Neural Networks, May 2000, 11(3)
- [71] Kohonen T, et al. Self organization of a massive document collection. IEEE Trans. on Neural Networks, May 2000, 11(3): 574 ~ 585

# 第 4 章 Hopfield 神经网络

## 4.1 概 述

人工神经网络的研究从 20 世纪 60 年代末开始进入十余年的沉寂时期,而从 20 世纪 80 年代初又开始蓬勃发展起来 出现了持续的研究高潮 直至今日。其中 J. J. Hopfield 的两篇经典论文<sup>[1,2]</sup>提出了 Hopfield 神经网络的概念,他和 D. Tank 将其用于优化问题时取得了令人瞩目的成果<sup>[3]</sup>,这成为这一研究领域的复兴和快速发展的最重要动因之一。

Hopfield 神经网络的基础构架是网络中每一个神经元的输出在乘以某些权值后送到其他所有神经元的输入端,从而形成一种全反馈结构,此外,每个神经元还各自接收一个外部输入。J. J. Hopfield 所提出的模型中网络按连续时间运行,一般称为连续时间 Hopfield 神经网络(简记为 HNN)或称为 Hopfield 模型(简记为 HM 目前学术界多采用此称呼,我们在下文中也用 HM 表示连续时间 Hopfield 神经网络)与 HM 对应的是按离散时间运行的 Hopfield 神经网络(简记为离散时间 HNN,下文中均取此种表示)。

一个 HM 的基本参数和运行规则包括:全部反馈权值、各神经元的外部输入(一般取定值)、各神经元的输入输出函数及其中的参数以及各神经元的输入(或输出)随时间变化的规律。一俟网络参数确定后,就可以按照某种定义,用这些参数和网络各神经元的输出求得一个 HM 的能量函数,它是描述该网络所处状态的重要参数。HM 的设计应保证,无论从何种初值出发,网络总能通过运行收敛到一个稳定状态,它相应于能量函数的某个局部极小点。

组合优化(combinatorial optimization)问题是一类应用很广的研究课题,其要求在于:满足一定的约束条件下求一组变量值,使某个目标函数达到极小。这一类问题一般是 NP 完备的或称为 NP 难度的(NP 是 non polynomial 的缩写),即求得最优解所需的计算量不是正比于变量数  $N$  的某个多项式(例如  $3N + N^2$ )而是正比于  $N$  的某个指数函数(例如  $e^N$ )。这样当  $N$  值增大时,为求得全局最优解,将出现计算量的指数式爆炸。因此,对于各种节约计算量而又能求得某个高质量次优解的高效算法的探寻备受关注。文献[3]将 HM 用于一个经典组合优化问题——TSP(traveling salesman problem)——的求解,取得了成果。其思路是将 TSP 的目标函数和约束条件同时映射为一个 HM 的能量函数,用网络中各神经元的输出表示各待求变量,当网络收敛到某个稳定解时,能量函数达到某个局部极小值。在一些特定的模拟实验中,通过仔细调节几个关键参数,使得此 HM 所收敛的稳定解是相应 TSP 的最优或次优可用解(可用是指该解满足约束条件)这一方案最

吸引人之处在于它是一个高度并行的求解算法并且有可能用硬件即 VLSI 芯片实现全部计算过程，从而在极短时间内求得所需的解。这是 HM 出现以来引起众多研究者关注的重要原因。

但是 HM 的研究进程并非一帆风顺，很多研究者在试图用 HM 来解决实际优化问题时遇到了困难，从而引起对于这一方案的怀疑与批评。这些批评集中在以下几点（可参见文献[4]、[5]）：（1）HM 中几个关键参数的确定非常困难，不同的问题需用不同的参数，而且要反复调试，稍有差池就不能得到可用解。有时甚至经过反复调试也找不到合适的参数值。（2）解的质量不高。由于目标函数的局部极小点的数量很多，HM 很可能收敛到一个距全局最优甚远的局部极小上。（3）即使 HM 对于解决 TSP 这一类目标函数的计算采取欧氏距离准则的问题时效果较好，对于不适用欧氏距离准则的问题则效果并不佳。（4）目前已有许多启发式优化算法可以用来解决实际优化问题。HM 与它们相比能操胜算吗？（5）HM 最初是针对 TSP 这个特定问题的，对于其他优化问题是否能采用 HM 呢？

在 HM 的研究遇到困难的同时，仍有一批研究者锲而不舍地在原 HM 的基础上不断改进。迄今为止，上述 1）、（2）两项问题已大体解决（见文献[4]给出的解决这一问题的参考文献）而（3）～（5）中列出的各项问题很多已有了正面的答案。很多模拟实验证明，对于许多实际优化问题，改进的 HM 无论在解的质量和求解的效率上都胜过目前常用的启发式算法。当然，还有不少问题正在研究之中，对于 HM 的改进仍在继续。改进的 HM 常称为 EHM（E 是 extended 的缩写），下文中我们均用 EHM 称呼各种改进的 HM。无论是经典的 HM 还是 EHM 其适用领域是 0-1 规划问题，TSP 也属于此领域。这是一个应用面甚宽的领域，它虽然隶属于范围更宽的整数规划问题，但是其中最重要的一部分（参考文献[6]）。

0-1 规划是求一组变量（其中每个变量的取值只能是 0 或 1），在满足若干线性等式和不等式约束的条件下，使某个目标函数达到极小值。下面给出 0-1 规划的定义。设有  $N$  个待求的变量  $x_i, i = 1 \sim N$ ，它们形成一个列向量  $\mathbf{X} = [x_1, x_2, \dots, x_N]^T$ 。设目标函数为  $F(\mathbf{X})$ ，用下式计算：

$$F(\mathbf{X}) = \frac{1}{2} \mathbf{X}^T \mathbf{Q} \mathbf{X} + \mathbf{g}^T \mathbf{X} \quad (4-1)^\text{①}$$

其中  $\mathbf{Q} = (q_{ij})$  是一个  $N \times N$  维对称实矩阵， $\mathbf{g} = [g_1, g_2, \dots, g_N]^T$  是一个  $N$  维实列向量。约束条件包括以下 3 项。

（1）满足下列  $M$  项等式约束（ $M < N$ ）

$$\mathbf{a}_m^T \mathbf{X} = b_m, \quad m = 1 \sim M \quad (4-2)$$

其中  $\mathbf{a}_m = [a_{m1}, a_{m2}, \dots, a_{mN}]^T$ 。

（2）满足下列  $K$  项不等式约束

$$\mathbf{r}_k^T \mathbf{X} \geq d_k \text{ 或 } \mathbf{r}_k^T \mathbf{X} \leq d_k, \quad k = 1 \sim K, d_k \geq 0 \quad (4-3)$$

其中  $\mathbf{r}_k = [r_{k1}, r_{k2}, \dots, r_{kN}]^T$ 。

① 目标函数也可表示为  $F(\mathbf{X}) = -\frac{1}{2} \mathbf{X}^T \mathbf{Q} \mathbf{X} - \mathbf{g}^T \mathbf{X}$ ，同一目标函数的两种表示法之差别仅在于各  $q_{ij}$  和  $g_i$  反号，对于算法的实质没有影响。

(3)  $x_i$  只能取值为 0 或 1 这表示为

$$x_i \in \{0,1\}, \quad \forall i = 1 \sim N \quad (4-4)$$

0-1 规划的目标即在于求得使  $F(\mathbf{X})$  达到极小且满足 3 项约束的解  $\mathbf{X}$  这可以表示为

$$\text{求可用解} \quad \mathbf{X} \rightarrow F(\mathbf{X}) = \min_{\mathbf{X} \text{ 可用}} F(\mathbf{X}) \quad (4-5)$$

其中可用解 (feasible solution) 表示  $\mathbf{X}$  满足 3 项约束。若  $F(\mathbf{X})$  有多个局部极小点 那么一个算法所求得的可解不一定是全局最优的。0-1 规划的任务即在于用尽可能少的计算量求得质量足够高的可用解 (与全局最优解的差距足够小)。

本章将讨论 HM 和各种 EHM 在 0-1 规划中的应用。4.2 节介绍经典 HM 将其用于 TSP 时所采取的技巧以及其局限性。4.3 节至 4.4 节将介绍三种 EHM 它们从不同角度出发改进并部分克服了经典 HM 的主要缺陷。给出许多具体应用实例的模拟实验结果。并且将指出若干尚待研究的问题。

离散时间 HNN 也是一种全反馈神经网络, 它与 HM 的主要区别在于网络的运作不是按照连续时间而是按照离散时间进行以及神经元的输入输出函数不同。这种网络主要用来模仿人脑的联想记忆 (associative memory) 功能。联想记忆是人类智能的一个非常重要的方面, 如何模仿、探究和利用这种功能无疑具有重大意义。离散时间 HNN 在若干方面与人脑的联想记忆功能有相似之处: (1) 这是一种并行分布式的存储机制, 其中不存在任何特定存储单元来存储某个特定的记忆项, 而所有记忆项都分布存储在网络中所有各对神经元之间的连接权中。(2) 各记忆项的存储 (或称为学习) 按照 Hebb 学习律进行<sup>[65]</sup>, 即相应于某个记忆项的任何两个神经元处于相同状态 (激发或抑制) 时, 它们之间的连接权值得得到加强, 反之便被削弱。(3) 记忆的提取是联想式的或称为按内容提取的, 后者简记为 CAM (content-addressed memory)。这就是说当输入一个原记忆项的缺损样本 (其中某些内容发生了缺失) 时仍能够正确地提取出原来存储的完整无缺的记忆项 (当然 缺损不能非常大)。离散时域 HNN 的研究除了有重要理论价值外, 在科技领域中也可以得到很多实际应用, 诸如人脸和声音的识别、图像的恢复和增强以及通信中的纠错编码、寻址等。由于篇幅限制, 本章将主要研究这种网络的理论问题。用时域离散 HNN 实现联想记忆时涉及的问题很多, 但最核心的内容是当网络神经元数  $N$  一定时, 网络的记忆容量和吸引域的确定及其尽可能的扩大 (有关精确定义见 4.5 节)。从 Hopfield 的早期研究工作发表<sup>[1~3]</sup> 至今, 这一领域的研究已有了长足的进展。在 Hopfield 早期工作的基础上, 对这种网络的运行机制所进行的理论分析和计算机模拟研究已使其容量和吸引域两方面都较早期的模型有了十分可观的改善。改进主要从改变神经元的输入输出函数和改变网络权矩阵的学习算法两方面着手。所使用的数学工具是概率和统计数学中常用的大数定律、中心局限定理等熟知的数学工具。由于问题比较复杂, 分析和计算十分繁杂和细致。在 4.5 节中将介绍上述的内容。基于 Hopfield 原型神经网络实现的联想记忆是一种自联想记忆, 记为 Auto-AM 即记忆项的提取者和被提取者是同一的。另外一种是异联想记忆, 记为 Hetro-AM, 或称为双向联想记忆 Bidirectional Associative Memory (BAM)。这时记忆项的提取和被提取者可以不一致。这将在 4.6 节讨论。在 4.6 节还将附带讨论一种可实现联想记忆功能的 BSB 神经网络。

## 4.2 HM 及其在 TSP 中的应用

HM 是由多个神经元构成的全反馈神经网络。它可以用一个模拟电路来实现并且有相应的连续时间运行方程。可以定义一个能量函数来描述系统所处的状态。网络的运行规则保证，无论何种初值出发，系统将运行并收敛到一个稳定平衡点——能量函数的局部极小点。将 TSP 的目标函数和约束条件映射为一个 HM 的能量函数，再转换为实际电路参数，则可以通过电路运行达到的收敛点求得所需的解。

### 4.2.1 HM 的电路结构和运行方程

设 HM 含  $N$  个神经元 编号为  $i = 1 \sim N$ 。神经元  $i$  的输出用  $x_i(t)$  表示 输入用  $u_i(t)$  表示，后者是由网络中所有神经元的输出反馈回来再加以外部输入后形成的， $t$  为连续时间。由于网络中每个神经元的结构和运行规则完全一致，只需给出其中任意第  $i$  个神经元

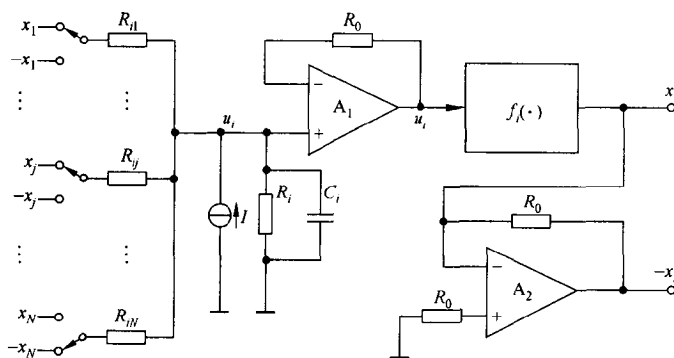


图 4-1 HM 的结构示意图

的结构即可窥系统之全豹，此结构如图 4-1 所示（其中  $I = z_i$  是外部输入恒流源）为简便起见，图 4-1 和下面讨论中  $x_i(t)$  和  $u_i(t)$  等的时间变量  $t$  皆略而不书。可以看到，结构中含两个运算放大器 A1 和 A2 以及一个函数变换器  $f_i(\cdot)$ 。A1 正端输入与输出值完全相同，皆用  $u_i$  表示。由于运放正端的输入阻抗可近似为无穷大值，A1 正端应满足电工理论中的节点电流方程：

$$\sum_{j=1}^N \frac{(\pm x_j - u_i)}{R_{ij}} + z_i - \frac{u_i}{R_i} - C_i \frac{du_i}{dt} = 0$$

其中  $x_j$  前的‘ $(\pm)$ ’表示相应的开关接到  $x_j$  时取‘+’号，接到  $-x_j$  时取‘-’号。如取  $C_i = 1$ ，做简单推导即得到

$$\frac{du_i}{dt} = -\frac{u_i}{\tau_i} + \sum_{j=1}^N w_{ij} x_j + z_i \quad (4-6)$$

其中

$$w_{ij} = (\pm) \frac{1}{R_{ij}}, \quad \tau_i = \left( \sum_{j=1}^N \frac{1}{R_{ij}} + \frac{1}{R_i} \right)^{-1}$$



由于电阻  $R_{ij}$  只能取正值, 反馈权值  $w_{ij}$  为正时与开关  $j$  相应的“(±)”取“+”, 为负时“(±)”取“-”。式(4-6)是 HM 各神经元的关键运行方程。

其次, 函数变换器将各  $u_i$  转换为  $x_i$ , HM 所用的转换函数是

$$x_i = f_i(u_i) = \frac{1}{2} \left[ 1 + \tanh\left(\frac{u_i}{T}\right) \right] \quad (4-7)$$

这是一个 Sigmoid 函数 参数  $T$  决定函数在  $u_i = 0$  附近的陡峭程度, 如图 4-2 所示。可以看到各  $x_i$  和  $u_i$  的变化范围是

$$x_i \in (0, 1), u_i \in (-\infty, +\infty), \quad \forall i = 1 \sim N$$

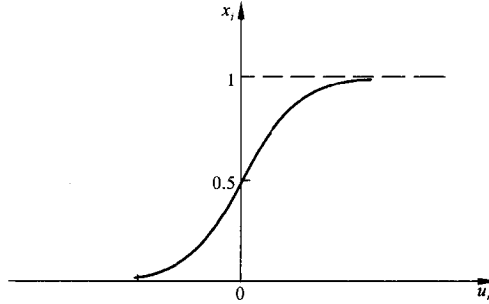


图 4-2 转换函数  $x_i = f_i(u_i)$

#### 4.2.2 HM 的能量函数和网络的稳定平衡解

令  $i = 1 \sim N$  由式(4-6)和式(4-7)定义了  $N$  个非线性联立微分方程。从任意初值  $x_i(0), i = 1 \sim N$  出发 需求得  $t \rightarrow \infty$  时各  $x_i(t)$  值 即系统的解。若  $t \rightarrow \infty$  时, 各  $x_i(t)$  趋向稳定不变之值, 则系统存在稳定平衡解。由于非线性联立微分方程没有一般的解析求解算法, 除了用硬件模拟电路求解外, 最直接有效的研究方法还是用数字计算机进行模拟求解。而从理论研究的角度出发, 则可以通过定义一个能量函数来回答稳定平衡解的有无、性质等一系列问题。在 HM 的研究和应用中, 式(4-6)右侧的各  $z_i$  都是不随时间改变的常数 且令各  $\tau_i$  一致 即  $\tau_i = \tau, \forall i = 1 \sim N$ 。设各  $x_i$  构成一个列向量  $\mathbf{X} = [x_1, x_2, \dots, x_N]^T$  则可定义相应于此  $\mathbf{X}$  的能量函数为

$$E(\mathbf{X}) = -\frac{1}{2} \mathbf{X}^T \mathbf{W} \mathbf{X} - \mathbf{Z}^T \mathbf{X} + \frac{1}{\tau} \sum_{i=1}^N \int_0^{x_i} f_i^{-1}(v) dv \quad (4-8)$$

其中  $\mathbf{Z} = [z_1, z_2, \dots, z_N]^T, \mathbf{W} = (w_{ij})$  是由各反馈权构成的  $N \times N$  维矩阵。设各  $w_{ij}, z_i$  皆为有限确定值,  $\tau > 0, T > 0$  且  $\mathbf{W}$  为对称阵。则易于证明  $E(\mathbf{X})$  是有下确界的函数,  $E(\mathbf{X})$  相对于  $t$  的导数可推导如下:

$$\frac{dE(\mathbf{X})}{dt} = \left( \frac{dE(\mathbf{X})}{d\mathbf{X}} \right)^T \left( \frac{d\mathbf{X}}{dt} \right) \quad (4-9)$$

其中

$$\frac{dE(\mathbf{X})}{d\mathbf{X}} = \left[ \frac{\partial E(\mathbf{X})}{\partial x_1}, \dots, \frac{\partial E(\mathbf{X})}{\partial x_N} \right]^T, \quad \frac{d\mathbf{X}}{dt} = \left[ \frac{dx_1}{dt}, \dots, \frac{dx_N}{dt} \right]^T$$

设  $U = [u_1, u_2, \dots, u_N]^T$  将式 (4-8) 代入式 (4-9) 并根据式 (4-7) 及  $W$  为对称阵 得到

$$\frac{dE(X)}{dX} = -WX + \frac{U}{\tau} - Z \quad (4-10)$$

注意由式 (4-6) 可以得到

$$\frac{dU}{dt} = WX - \frac{U}{\tau} + Z \quad (4-11)$$

其中  $\frac{dU}{dt} = \left[ \frac{du_1}{dt}, \frac{du_2}{dt}, \dots, \frac{du_N}{dt} \right]^T$ 。与式 (4-10) 对比, 即得

$$\frac{dE(X)}{dX} = -\frac{dU}{dt} \quad (4-12)$$

设有下列对角阵

$$\frac{dU}{dX} = \begin{bmatrix} \frac{du_1}{dx_1} & & 0 \\ & \frac{du_2}{dx_2} & \\ & & \ddots \\ 0 & & & \frac{du_N}{dx_N} \end{bmatrix}, \quad \frac{du_i}{dx_i} = \frac{df_i^{-1}(x_i)}{dx_i} > 0, \quad \forall i = 1 \sim N \quad (4-13)$$

则有

$$\frac{dU}{dt} = \frac{dU}{dX} \cdot \frac{dX}{dt} \quad (4-14)$$

将式 (4-14) 代入式 (4-12) 再代入式 (4-9) 且由式 (4-7) 可知  $f_i^{-1}(x_i)$  非降 即得

$$\frac{dE(X)}{dt} = -\left(\frac{dX}{dt}\right)^T \left(\frac{dU}{dX}\right) \left(\frac{dX}{dt}\right) = -\sum_{i=1}^N \left(\frac{dx_i}{dt}\right)^2 \frac{df_i^{-1}(x_i)}{dx_i} \leq 0 \quad (4-15)$$

由式 (4-15) 可以导出以下两点结论。

(1) 由于  $E(X)$  对时间的导数恒非正且  $E(X)$  有确下界 因此无论从何种初值出发以及无论输入  $Z$  取何值 随着  $t \rightarrow \infty$  必有  $dE(X)/dt \rightarrow 0$ 。即系统将收敛于  $E(X)$  的一个局部极小点或全局最小点。

(2) 因为式 (4-15) 最右侧第 2 项  $df_i^{-1}(x_i)/dx_i > 0$  恒成立 若  $dE(X)/dt = 0$  则必有  $dx_i/dt = 0, \forall i = 1 \sim N$ , 即每个输出恒定不变 系统有稳定解。

### 4.2.3 HM 能量函数的进一步讨论

按照式 (4-8) 定义的能量函数, 由于存在右侧第 3 项, 因此不是 Liapunov 函数, 从而引起很多运算上的困难和问题。因此 J. J. Hopfield 以及其他所有研究者都将这一项去除而采取下列的能量函数定义:

$$E(X) = -\frac{1}{2}X^T W X - Z^T X \quad (4-16)$$

为了不脱离硬件电路实现的背景, Hopfield 提出将该项去除的前提是式 (4-7) 中的参数  $T$  取得足够小, 这样函数  $f_i(\cdot)$  非常接近于阶跃式的函数,  $f_i^{-1}(v)$  在  $[0, 1]$  区间中非常接近

于 0，因此式 (4-8) 最右侧的积分项可略去。

后来的一些研究者，从一开始就不顾电路背景而直接按式 (4-16) 来定义  $E(\mathbf{X})$  或定义为

$$E(\mathbf{X}) = \frac{1}{2} \mathbf{X}^T \mathbf{W} \mathbf{X} + \mathbf{Z}^T \mathbf{X} \quad (4-17)$$

无论哪一种定义都给出了 Liapunov 函数。然后再从该能量函数出发来定义系统的运行规则，

$$\frac{dU}{dt} = - \frac{dE(\mathbf{X})}{d\mathbf{X}} \quad (4-18)$$

按照式 (4-16) 的定义即可导出

$$\frac{dU}{dt} = \mathbf{W} \mathbf{X} + \mathbf{Z} \quad (4-19)$$

反之，按照式 (4-17) 的定义即可导出

$$\frac{dU}{dt} = -\mathbf{W} \mathbf{X} - \mathbf{Z} \quad (4-20)$$

在实际运算以及解决优化问题时，无论取哪一种定义都是可行的。而式 (4-7) 总是决定了  $\mathbf{U}$  至  $\mathbf{X}$  的转换函数，在没有电路背景约束的条件下，其中的参数  $T$  可取为任意大于 0 的实数。

#### 4.2.4 能量函数为 Liapunov 函数时系统的解

当  $E(\mathbf{X})$  按式 (4-16) 或式 (4-17) 的定义成为 Liapunov 函数时，式 (4-15) 依然成立，因而 4.2.2 小节给出的两点结论仍然成立，即当  $t \rightarrow \infty$  时  $E(\mathbf{X})$  收敛到它的一个局部极小点或全局最小点。在此点上  $dx_i/dt = 0, \forall i = 1 \sim N$  因此它是系统的平衡点。下面继续研究解的性质。设有下列对角阵：

$$\frac{d\mathbf{X}}{d\mathbf{U}} = \begin{bmatrix} \frac{dx_1}{du_1} & & 0 \\ & \frac{dx_2}{du_2} & \\ & & \ddots \\ 0 & & & \frac{dx_N}{du_N} \end{bmatrix} \quad (4-21)$$

则可得

$$\frac{d\mathbf{X}}{dt} = \left( \frac{d\mathbf{X}}{d\mathbf{U}} \right) \left( \frac{d\mathbf{U}}{dt} \right) \quad (4-22)$$

根据式 (4-7),  $dx_i/du_i = 2x_i(1-x_i), \forall i = 1 \sim N$ ，再引用与式 (4-17) 定义的  $E(\mathbf{X})$  相对应的式 (4-20) (也可引与式 (4-16) 定义的  $E(\mathbf{X})$  相对应的式 (4-19) 此处只引其一) 即得到

$$\frac{d\mathbf{X}}{dt} = - \begin{bmatrix} 2x_1(1-x_1) & & 0 \\ & 2x_2(1-x_2) & \\ & & \ddots \\ 0 & & & 2x_N(1-x_N) \end{bmatrix} (\mathbf{W} \mathbf{X} + \mathbf{Z}) \quad (4-23)$$

如果  $\tilde{\mathbf{X}}$  是系统的解(平衡点)则应满足

$$\left. \frac{d\mathbf{X}}{dt} \right|_{\mathbf{X}=\tilde{\mathbf{X}}} = \mathbf{0} \quad (4-24)$$

设  $\tilde{\mathbf{X}} = [\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_N]^T$  由式(4-23)和式(4-24)可知,满足下列3个项目中的任意一项的  $\tilde{\mathbf{X}}$  为平衡点<sup>[10]</sup>(还有其他很多平衡点见4.4节)。

$$(1) \tilde{x}_i \in \{0, 1\}, \quad \forall i = 1 \sim N \quad (4-25a)$$

$$(2) \mathbf{W}\mathbf{X} + \mathbf{Z} = \mathbf{0}, \quad 0 < \tilde{x}_i < 1, \quad \forall i = 1 \sim N \quad (4-25b)$$

$$(3) \tilde{x}_i \in \{0, 1\}, \quad \forall i = 1 \sim N \quad \text{且} \quad \mathbf{W}\mathbf{X} + \mathbf{Z} = \mathbf{0} \quad (4-25c)$$

那么满足上述3项的各种解中,哪一些是稳定平衡解呢?显然HM所收敛的解只能是稳定平衡点而非不稳定平衡点。下面对于式(4-25)给出的各平衡点进行分析。

首先,对于2)给出的平衡点,由式(4-25b)可知,若  $\mathbf{W}$  非奇则这样的平衡点可能存在且最多只可能出现一个,即

$$\mathbf{X} = -\mathbf{W}^{-1}\mathbf{Z}, \quad \text{且} \quad 0 < \tilde{x}_i < 1, \quad i = 1 \sim N \quad (4-26)$$

上文中“最多”的意思是指由式(4-26)第1个公式解出的  $\tilde{\mathbf{X}}$  的各分量  $\tilde{x}_i$  如不满足第2个公式的不等式约束,则此平衡点仍不存在。此平衡点是否稳定取决于矩阵  $\mathbf{W}$  是否正定。若  $\mathbf{W}$  为正定阵,换言之  $\mathbf{W}$  的所有特征值皆大于0,则式(4-26)所确定的  $\tilde{\mathbf{X}}$  是一个稳定平衡解,反之则否(限于篇幅,不做证明)应指出,用HM求解0-1规划问题时,解必须满足4.1节给出的条件(3)(式(4-4)),即各  $\tilde{x}_i$  应满足式(4-25a)。这时  $\tilde{\mathbf{X}}$  应是一个  $N$  维单位超立方体的角点。而式(4-26)即使给出了一个稳定平衡解,它也是这个立方体的“内点”,因此是无用解。

其次,对于1)、(3)给出的平衡解中,分析哪些是稳定平衡解。这些解相应于  $N$  维单位超立方体的各个角点,这种角点共有  $2^N$  个,它们都符合4.1节给出的条件(3)。因此只要某个角点是稳定平衡点,它就是一个可用解。现在从式(4-23)、式(4-24)和式(4-25a)出发来讨论这个问题。设  $\tilde{\mathbf{X}}$  是某个角点,则  $\tilde{\mathbf{X}}$  必满足式(4-24)和式(4-25a)。设  $\mathbf{X} = \tilde{\mathbf{X}} + \mathbf{X}'$ ,  $\mathbf{X}' = [x'_1, x'_2, \dots, x'_N]^T$ ,  $0 < |x'_i| \ll 1$  即  $\mathbf{X}$  与  $\tilde{\mathbf{X}}$  之间有微小的偏离,  $\mathbf{X}'$  是一个偏离变向量。设

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_1 \\ \mathbf{W}_2 \\ \vdots \\ \mathbf{W}_N \end{bmatrix}, \quad \mathbf{W}_i = [w_{i1}, w_{i2}, \dots, w_{iN}], \quad i = 1 \sim N \quad (4-27)$$

已知  $\mathbf{Z} = [z_1, z_2, \dots, z_N]^T$ , 利用以上各公式作简单推导,即可求得(略去二阶微小量)

$$\left[ \frac{d\mathbf{X}}{dt} \right]_{\mathbf{X}=\tilde{\mathbf{X}}+\mathbf{X}'} = \begin{bmatrix} 2(2\tilde{x}_1 - 1)(\mathbf{W}_1\tilde{\mathbf{X}} + z_1) & & 0 \\ & \ddots & \\ 0 & & 2(2\tilde{x}_N - 1)(\mathbf{W}_N\tilde{\mathbf{X}} + z_N) \end{bmatrix} \mathbf{X}' \quad (4-28)$$

由于  $\mathbf{X}$  是一固定向量,  $\mathbf{X}'$  是一个变向量,所以上式可写成

$$\left[ \frac{d\mathbf{X}'}{dt} \right] = \begin{bmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ & & \ddots \\ 0 & & & \lambda_N \end{bmatrix} \mathbf{X}', \quad \lambda_i = 2(2\tilde{x}_i - 1)(\mathbf{W}_i \tilde{\mathbf{X}} + \mathbf{z}_i), \quad i = 1 \sim N \quad (4-29)$$

式中的  $\lambda_i, i = 1 \sim N$  称为 HM 系统在  $\tilde{\mathbf{X}}$  的特征值 (注意, 它们是系统的特征值, 而不是矩阵  $\mathbf{W}$  的特征值) 若  $\lambda_i < 0, \forall i = 1 \sim N$  则此  $\tilde{\mathbf{X}}$  是一个稳定平衡点。这是因为, 若某个偏离分量  $x'_i > 0$  则  $dx'_i/dt = \lambda_i x'_i < 0$  这时此偏离将被自动纠正 若  $x'_i < 0$  同样被纠正。这样, 可以给出下列结论:

若  $\tilde{\mathbf{X}}$  是 HM 系统  $N$  维单位超立方体的一个角点, 在满足下列条件时,  $\tilde{\mathbf{X}}$  是稳定平衡解:

$$\lambda_i = 2(2\tilde{x}_i - 1)(\mathbf{W}_i \tilde{\mathbf{X}} + \mathbf{z}_i) < 0, \quad i = 1 \sim N \quad (4-30)$$

若矩阵  $\mathbf{W}$  的主对角元素皆为 0 即满足

$$w_{ii} = 0, \quad i = 1 \sim N \quad (4-31)$$

这时各稳定平衡角点  $\tilde{\mathbf{X}}$  与它的各个  $i$  分量互异的相邻角点  $\tilde{\mathbf{X}}^{(i)} = [\tilde{x}_1^{(i)}, \tilde{x}_2^{(i)}, \dots, \tilde{x}_N^{(i)}]^T$  (即  $\tilde{x}_i^{(i)}$  与  $\tilde{x}_i$  互异, 而二者的其他分量相同) 之间有如下关系:

(1)  $\tilde{\mathbf{X}}^{(i)}, i = 1 \sim N$  都是非稳定平衡角点

(2)  $E(\mathbf{X}) < E(\mathbf{X}^{(i)}), i = 1 \sim N$  (4-32)

首先证明第 1) 项。已知 HM 在  $\tilde{\mathbf{X}}$  点的第  $i$  特征值  $\lambda_i$  用式 (4-30) 计算 而在  $\tilde{\mathbf{X}}^{(i)}$  点的第  $i$  特征值  $\lambda_i^{(i)}$  可用下式计算:

$$\lambda_i^{(i)} = 2(2\tilde{x}_i^{(i)} - 1)(\mathbf{W}_i \tilde{\mathbf{X}}^{(i)} + \mathbf{z}_i)$$

由于  $\tilde{x}_i^{(i)}$  与  $\tilde{x}_i$  互异 即前者为 0 则后者 1 或反之 因此  $(2\tilde{x}_i^{(i)} - 1) = -(2\tilde{x}_i - 1)$ 。此外 因为  $w_{ii} = 0$  且  $\tilde{\mathbf{X}}^{(i)}$  与  $\tilde{\mathbf{X}}$  只有第  $i$  个分量互异 所以  $\mathbf{W}_i \tilde{\mathbf{X}}^{(i)} + \mathbf{z}_i = \mathbf{W}_i \tilde{\mathbf{X}} + \mathbf{z}_i$ 。这样 立即可知

$$\lambda_i^{(i)} = -\lambda_i$$

而  $\tilde{\mathbf{X}}$  是稳定平衡点 故  $\lambda_i < 0$  因此  $\lambda_i^{(i)} > 0$ 。易于证明 任何平衡点 其 HM 系统特征值只要有一个大于 0 则必不稳定 所以  $\tilde{\mathbf{X}}^{(i)}$  是非稳定平衡角点。对任何  $i = 1 \sim N$  此论证皆成立。

其次证明第 2) 项。对于任何  $i$  值  $\tilde{\mathbf{X}}$  与  $\tilde{\mathbf{X}}^{(i)}$  之间只有第  $i$  分量不同, 因此只需考虑  $x_i$  变至  $\tilde{x}_i^{(i)}$  时相应  $E(\mathbf{X})$  的变化即可。为此设一只有第  $i$  维变化的偏移向量  $\mathbf{X}'(i) = [0, 0, \dots, 0, x'_i, 0, \dots, 0]^T$  若  $\tilde{x}_i = 1$ , 令  $x'_i > 0$ ; 若  $\tilde{x}_i = 0$ , 令  $x'_i < 0$ 。由式 (4-17)、式 (4-22) 和式 (4-29) 可以导出

$$E(\tilde{\mathbf{X}} + \mathbf{X}'(i)) - E(\tilde{\mathbf{X}}) \propto -\lambda_i (x'_i)^2 \quad (4-33a)$$

$$E(\tilde{\mathbf{X}}^{(i)} - \mathbf{X}'(i)) - E(\tilde{\mathbf{X}}^{(i)}) \propto -\lambda_i^{(i)} (x'_i)^2 \quad (4-33b)$$

由于  $\lambda_i < 0$  当  $\mathbf{X}$  由  $\tilde{\mathbf{X}}$  向  $\tilde{\mathbf{X}}^{(i)}$  方向移动时  $E(\mathbf{X})$  增加; 由于  $\lambda_i^{(i)} > 0$  当  $\mathbf{X}$  由  $\tilde{\mathbf{X}}^{(i)}$  向  $\tilde{\mathbf{X}}$  方向移动时  $E(\mathbf{X})$  减少。因此当  $\mathbf{X}$  由  $\tilde{\mathbf{X}}$  移至  $\tilde{\mathbf{X}}^{(i)}$  时  $E(\mathbf{X})$  只可能单调增加, 即证实式 (4-32) 成立。之所以只可能出现单调增加的理由是如果出现增—降—增的格局, 那么在这两个相邻角点之间将出现两个  $E(\mathbf{X})$  相对于  $x_i$  导数的 0 点 但是  $E(\mathbf{X})$  只是  $x_i$  的二次函数 它相

对于  $x_i$  的导数不可能出现两个 0 点，因此这种情况不会出现。

从式 (4-33) 和以上论证出发，可以得到下列逆向结论：如果  $E(\tilde{\mathbf{X}}) < E(\tilde{\mathbf{X}}^{(i)})$   $i = 1 \sim N$  成立 那么必然有  $\lambda_i < 0, i = 1 \sim N$ 。事实上这两个条件完全等价。由此可得如下结论：

若  $\tilde{\mathbf{X}}$  是 HM 系统  $N$  维单位超立方体的一个角点，在满足下列条件时， $\tilde{\mathbf{X}}$  是稳定平衡解：

$$E(\tilde{\mathbf{X}}) < E(\tilde{\mathbf{X}}^{(i)}), \quad i = 1 \sim N$$

此式即式 (4-32)。

以上讨论的立足点是矩阵  $\mathbf{W}$  的主对角元素为 0 即式 (4-31) 必须成立。如果此条件不满足，可以采取下列技术使之满足而又不影响所需达到的目标。由式 (4-17) 可见，一个 HM 的能量函数中与  $w_{ii}$  有关的是下列项：

$$\frac{1}{2} \sum_{i=1}^N w_{ii} x_i^2$$

当 HM 收敛到一稳定平衡角点时， $x_i = 0$  或 1 这时在上式中用  $x_i$  替代  $x_i^2$  并不会使结果有任何变化。所以 当式 (4-31) 原本不成立的情况下，只需要将矩阵  $\mathbf{W}$  的主对角元素皆用 0 取代，而将向量  $\mathbf{Z}$  的各分量  $z_i$  用  $z_i + w_{ii}$  取代，这时在各个角点上由式 (4-17) 或式 (4-16) 给出的  $E(\mathbf{X})$  不会产生任何变化。HM 即可按此修正后的  $\mathbf{W}$  和  $\mathbf{Z}$  运行。

现在讨论上述策 (3) 项的平衡点 (式 4-25c)。如果此解存在 按照前文的分析 最多只能有一个，而且必定是准稳定的。这是因为此平衡点的各个特征值  $\lambda_i = 0$  这时  $\mathbf{X}$  与各邻点  $\tilde{\mathbf{X}}^{(i)}$  没有能量差异，因此 HM 系统最终收敛的解不能确定是其中哪一个。

最后应指出，以上讨论都基于式 (4-17) 定义的  $E(\mathbf{X})$ ，如果基于式 (4-16) 定义的  $E(\mathbf{X})$  时可作类似讨论，只是式 (4-29) 式 (4-30) 中  $\lambda_i = -2(2\tilde{x}_i - 1)(\mathbf{W}_i \tilde{\mathbf{X}} + \mathbf{z}_i)$ 。

#### 4.2.5 HM 用于 TSP 的求解

设有  $M$  个城市 记为  $C_1, C_2, C_3, \dots, C_M$ 。 $C_1, C_2$  之间的距离记为  $d_{12}$ ， $C_2, C_3$  之间的距离记为  $d_{23}$  等。有一旅行商试图从其中某城市出发，走一条最短路径遍访所有其他城市后回到出发点，且规定每个城市只能访问一次。这样，问题归结为求一条穿过所有  $M$  个城市且每个城市只能通过一次的最短闭合路径。在各城市之间的距离已给定的条件下，求此最小路径的问题即是 TSP。图 4-3 给出了一个  $M = 10$  时  $C_1 \sim C_{10}$  在平面图上分布的例子，由此图可以确定任意两个城之间的欧氏距离并以其之作为两城市之间距离的定义。当  $M = 10$  时 存在  $\frac{1}{2}(10! - 1) = 1814400$  种不同的闭合路径 例如

$C_1 \rightarrow C_5 \rightarrow C_7 \rightarrow C_4 \rightarrow C_6 \rightarrow C_9 \rightarrow C_8 \rightarrow C_2 \rightarrow C_3 \rightarrow C_{10} \rightarrow C_1$  构成一条闭合路径，其长度为  $d = d_{15} + d_{57} + d_{74} + d_{46} + d_{69} + d_{98} + d_{82} + d_{23} + d_{3,10} + d_{10,1}$ 。

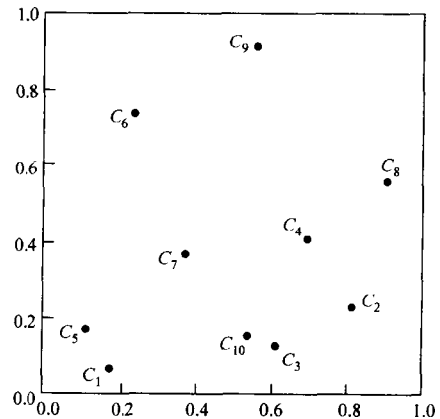


图 4-3  $C_1 \sim C_{10}$  的平面分布图

如果用枚举法从这 180 万余种路径中选出一最短者，工作量将大得惊人，遑论  $M$  更大的情形。下面介绍用 HM 来求解。

为简单起见，用  $M = 5$  的情况为例来进行讨论。首先，构造一个由 5 行 5 列构成的阵列如下面表 4-1 所示。阵列中的各行  $m = 1 \sim 5$  依次表示城市  $C_1 \sim C_5$ ，各列  $i = 1 \sim 5$  依次表示按顺时针或反时针计算时一条路径的编号。在表 4-1 中第 1 行第 2 列元素为 1 而其他为 0，表明  $C_1$  出现在路径的第 2 位，余可类推。这样表 4-1 就定义了一条闭合路径  $C_3 \rightarrow C_1 \rightarrow C_5 \rightarrow C_2 \rightarrow C_4 \rightarrow C_3$ 。由于 1 个城市只能出现在路径的一个位置上，所以阵列中每行只有 1 个元素为 1 而其他为 0。同样，路径的任一位置只允许有一城市，所以每列只允许有一个元素为 1 而其他为 0。在全部阵列中只允许有  $M$  个 1（此例中  $M = 5$ ）而其他元素皆为 0。这样，可以用一个双下标变量  $x_{mi}$  来表示表 4-1 阵列中的各个元素，若  $x_{mi} = 1$  表明城市  $C_m$  在路

表 4-1  $M = 5$  时的路径阵列

	$m \backslash i$	1	2	3	4	5
$C_1$	1	0	1	0	0	0
$C_2$	2	0	0	0	1	0
$C_3$	3	1	0	0	0	0
$C_4$	4	0	0	0	0	1
$C_5$	5	0	0	1	0	0

径中的第  $i$  位，若  $x_{mi} = 0$  则否。对于任何  $M$  个城市的 TSP 都可以按此要求定义  $M^2$  个双下标变量  $x_{mi}$ ， $m = 1 \sim M$ ， $i = 1 \sim M$ 。如果它们满足下列约束，就定义了惟一的一条闭合路径：

$$\sum_{i=1}^M x_{mi} = 1, \forall m = 1 \sim M, \sum_{m=1}^M x_{mi} = 1, \forall i = 1 \sim M, x_{mi} \in \{0, 1\} \quad (4-34)$$

全路径的长度  $d$  可用下式计算：

$$d = \sum_{i=1}^M \sum_{m,n=1}^M d_{mn} x_{mi} (x_{n,i+1} + x_{n,i-1}) \quad (4-35)$$

注意  $d_{mm} = 0$  并且式中的下标是按模  $M$  数计算的，其规则为：若  $i = 0$  则用  $M$  代之，若  $i = M + 1$  则用 1 代之，其他情况保持不变。这样 TSP 归结为在约束条件式 (4-34) 得到保证的前提下，求  $M^2$  个变量  $x_{mi}$  使得  $d$  达最小。TSP 还可以选择其他的变量来定义路径，从而形成各种不同的约束条件和目标函数并使得求解算法各异且利弊各异<sup>[7]</sup>。下面给出的是文献[3] 中的求解方案。

对于  $M$  个城市的 TSP 用具有  $M^2$  个神经元的 HM 来解决，每个神经元的输出、输入皆用双下标表示为  $x_{mi}$ ， $u_{mi}$ ， $m, i = 1 \sim M$ 。网络的反馈权相应地表示为  $w_{mi,nj}$ ，外部输入表示为  $z_{mi}$ 。如果将  $M^2$  个  $x_{mi}$  排列成一个  $M^2$  维列向量  $\mathbf{X}$ ，将  $M^2$  个  $z_{mi}$  排列成列向量  $\mathbf{Z}$ ，则可以用式 (4-16) 来定义此 HM 的能量函数  $E(\mathbf{X})$ ，其中  $\mathbf{W}$  为  $M^2 \times M^2$  维反馈权矩阵。将此式展开，得到

$$E(\mathbf{X}) = -\frac{1}{2} \sum_{m,i=1}^M \sum_{n,j=1}^M x_{mi} w_{mi,nj} x_{nj} - \sum_{m,i=1}^M z_{mi} x_{mi} \quad (4-36)$$

TSP 作为一个 0-1 规划问题，就是要找到满足式 (4-34) 约束的可用解  $\mathbf{X}$ ，使得按照式

(4-35) 计算的目标函数  $F(\mathbf{X}) = d$  达到最小。这需要将约束条件和目标函数都映射到  $E(\mathbf{X})$ ，可通过求得使  $E(\mathbf{X})$  达到极小的解  $\mathbf{X}$  来同时满足上述两项要求。Hopfield 和 Tank<sup>[3]</sup> 采取的映射方案是令  $E(\mathbf{X})$  由下式计算：

$$E(\mathbf{X}) = \frac{\alpha}{2} \sum_{m=1}^M \sum_{\substack{i,j=1 \\ i \neq j}}^M x_{mi} x_{mj} + \frac{\beta}{2} \sum_{i=1}^M \sum_{\substack{m,n=1 \\ m \neq n}}^M x_{mi} x_{ni} + \frac{\zeta}{2} \left( \sum_{m=1}^M \sum_{i=1}^M x_{mi} - M \right)^2 + \frac{\gamma}{2} F(\mathbf{X}) \quad (4-37)$$

此式右侧第 1 项保证表 4-1 每行中 1 的个数不超过 1 时，该项达到最小值 0；右侧第 2 项保证每列中 1 的个数不超过 1 时达到最小值 0；右侧第 3 项保证各行各列中 1 的个数为  $M$  时达到最小值 0。如果这三项都达到 0 值，则约束条件式 (4-34) 的前两部分都得到了满足。 $\alpha, \beta, \zeta, \gamma$  是 4 个适当选择的正的常数， $F(\mathbf{X})$  用式 (4-35) 计算。对比式 (4-36) 和式 (4-37) 可以知道 HM 的各反馈权值  $w_{mi,nj}$  和各外部输入  $z_{mi}$  可用下式计算：

$$w_{mi,nj} = -\alpha \delta_{mn} (1 - \delta_{ij}) - \beta \delta_{ij} (1 - \delta_{mn}) - \zeta - \gamma d_{mn} (\delta_{j,i+1} + \delta_{j,i-1}) \quad (4-38)$$

$$z_{mi} = \zeta M \quad \forall m, i, n, j = 1 \sim M$$

其中

$$\begin{aligned} \delta_{mn} &= 1, \quad m = n, \quad \sigma_{ij} = 1, \quad i = j \\ \delta_{ij} &= 1, \quad i = j \\ \delta_{mn} &= \delta_{ij} = 0, \quad \text{其他} \end{aligned}$$

文献 [3] 和 [8] 针对式 (4-38) 的权和外输入信号取值以  $M = 10$  并按照图 4-3 中给出的 10 个城市为例进行了模拟实验。HM 的运行方程由式 (4-6) 确定，由  $u_i$  至  $x_i$  的转换按式 (4-7) 进行。所用的各项参数是  $\alpha = \beta = 500, \zeta = 200, \gamma = 500, \tau = 1, T = 0.02$ 。各个城市之间的距离  $d_{mn}$  为图 4-3 中  $C_m$  与  $C_n$  之间的几何距离。模拟实验中取 20 组不同的初值来运行网络并检验求解的结果。每一组初值  $u_{mi}(0), m, i = 1 \sim M$  是按照下式求得的：

$$u_{mi}(0) = u_0 + \delta u_{mi}$$

其中  $\delta u_{mi}$  是在  $[-0.1T, 0.1T]$  区间内具有均匀分布的随机量。 $u_0$  的选择应使下式得到满足：

$$\sum_{m,i=1}^M x_{mi}(0) = \sum_{m,i=1}^M f_i(u_{mi}(0)) = 10$$

即  $u_0$  的取值为  $-1.0986$ 。

这 20 组实验的结果是，其中大部分的实验经过若干次迭代即收敛于符合式 (4-34) 约束条件的稳定平衡解，从而得到了一条闭合路径。这些路径中包含最优解（最短路径），也包含若干接近最优解的次优解，例如：

$$C_{10} \ C_3 \ C_2 \ C_4 \ C_8 \ C_9 \ C_6 \ C_7 \ C_5 \ C_1 \ C_{10} (d = 2.71, \text{最优解});$$

$$C_6 \ C_9 \ C_8 \ C_4 \ C_2 \ C_3 \ C_{10} \ C_7 \ C_1 \ C_5 \ C_6 (d = 2.83, \text{次优解}).$$

考虑到用枚举法求解此问题时需进行约 180 万次计算才能找到最优解，而在这个采用 HM 的实验中仅做了 20 次实验就求得了最优解及若干高质量的次优解，效率的提高几乎达到  $10^5$ 。当  $M$  进一步增大时，效率的提高将更可观。这一实验结果给予人们的初步印象是很深的。

#### 4.2.6 HM 用于 TSP 求解的进一步讨论

虽然 Hopfield 等的初步实验结果使人印象颇深，但随后，其他研究者却发现了很多



问题，这些问题可归结如下。

(1) 用 HM 来解决上述 10 个城市及特定的城市配置 (图 4-3) TSP 时 运算方程中的参数  $T$  及定义能量函数所用的  $\alpha, \beta, \zeta, \gamma$  等参数都是经过精心挑选和反复试验后才择定的。例如参数  $T$ ，若选得太大则所收敛的解不能保证约束条件  $x_{m,i} \in \{0,1\}$  的实现 这时解不可用；若选得太小，则所收敛的解质量相当差（即路径长度比最优解长得多）。 $\alpha, \beta, \zeta, \gamma$  的选择也十分困难。问题在于，所有这些参数的选择是与待解决的问题有关的，即对不同的  $M$  值和不同的城市位置分布需选择不同的参数才有可能得到最佳或质量较高的次最优解。这样，用凑试法反复实验来找到合适的参数成为一个非常困难的任务，甚至没有保证必然会找到合适参数。这就大大阻碍了 HM 在实际优化问题中的应用。显然，需要人工神经网络实现的是一套系统的 0-1 规划算法 即在 4.1 节中所述的满足式 (4-2) ~ 式 (4-4) 这 3 个约束条件下求最优的  $\mathbf{X}$  使目标函数  $F(\mathbf{X})$  达到最小 (式 4-5)) 且其中涉及的参数可按系统方法求出而非凑试出来的。这套算法还应接受各种实际优化问题检验并与其他已有的启发式 0-1 规划算法比较，以考察其是否的确有效。这就是 20 世纪 80 年代中期 HM 出现以后很多研究者孜孜以求的目标，从而出现了各种 EHM。

(2) 可以看到，约束条件分成两组。式 (4-2) 和式 (4-3) 是一组 它们要求  $\mathbf{X}$  各分量的一些线性组合满足若干等式和不等式。式 (4-4) 是另一组，它要求最终获得的稳定解  $\mathbf{X}$  的各个分量  $x_i$  只能取值为 0 或 1。只有两方面要求都满足的解才是可用解。

为了满足前一组约束，可以采取的方案之一是约束平面法，即将式 (4-2) 和式 (4-3) 规定的条件转化为一个约束超平面，当向量  $\mathbf{X}$  (或相应的扩展向量) 在此超平面中时，各项约束自动满足，然后在此超平面中求得使  $F(\mathbf{X})$  最小的解。这一方法将在 4.3 节中讨论。另外可取的一种方案是罚函数法，即令能量函数  $E(\mathbf{X})$  由目标函数  $F(\mathbf{X})$  和反映约束满足程度的一个罚函数之和构成，二者各乘以适当的正系数。当  $E(\mathbf{X})$  达到其局部或全局极小时，罚函数趋向于 0，从而使约束得到满足且使  $F(\mathbf{X})$  达到一局部或全局极小值。其中的正系数应该用系统的方法而非凑试法求得。此方案将在 4.4 节中讨论。实际上，本节介绍的 Hopfield 和 Tank 用 HM 求 TSP 解的方案就是一种罚函数法，其缺点是涉及的罚函数项及相应的系数太多且没有一种系统地确定这些系数的方法。此外，该方法也不能推广到一般的含有等式及不等式约束的 0-1 规划问题。

而满足后一组约束，即  $x_i \in \{0,1\}, \forall i = 1 \sim N$  则可能遇到另一些困难。当采用 4.3 节所述的约束超平面方法时，可能出现的一个问题是在该平面中搜索  $F(\mathbf{X})$  的最小值但系统的稳定平衡点不再是  $N$  维单位超立方格的某些角点而是此平面与立方格的某些“交点”。如果这些交点都是角点则不成问题，但是有些交点不是角点时，系统所收敛的解就有可能不满足这后一组约束，从而成为不可用解。为克服此困难，可以采用 4.3 节将要叙述的一种修正方法——Tabu 搜索方案。对于各种实际优化问题，其有效性尚待检验，是否有效率更高的方案也尚待研究。对于 4.4 节所述的罚函数法也存在着满足前一组约束较易实现而满足后一组约束比较困难的问题。其困难与约束平面法不同，问题出在所收敛的解的若干分量既不很接近 0 又不很接近 1，从而很难对其作出明确判断，这样解就不可用。当存在不等式约束时，问题更困难些。目前虽然已提出了一些改进方案 (见 4.4 节) 但迄今为止仍是一个尚在探索的问题。

(3)关于用 HM 求解 TSP 及其他 0-1 规划问题时,所得可用解的质量。因为目标函数  $F(\mathbf{X})$  (见式 (4-1)) 是二次函数 其局部极小点的个数很多 而 HM 的运行算法是最陡下降算法,因此所求得解只是某个局部极小解,其质量不一定很高。为解决此问题,所采取的策略通常是在权的迭代计算中引入模拟退火(SA)或登山(hill-climbing,HC)算法。例如,令式(4-7)中的参数  $T$  为可变的,当迭代计算开始时  $T$  取很大值,而当迭代接近结束时, $T$  逐渐变为很小值。另一种方法是将权调整的最陡下降算法变为登山算法,即在迭代开始时权的调整方向可在  $E(\mathbf{X})$  升、降两种方向中按相同概率随机选择 而随着迭代进展使降的概率渐增、增的概率渐减,直至最后只有降的方向。有关问题可见 4.3 节的详细讨论。

(4)提高 HM 所求解的质量的方法还很多,例如:弹性网络模型(elastic net model)<sup>[11,12,13]</sup> 分解模型(deformable model)<sup>[14]</sup>,自组织特征映射(SOFM)<sup>[15~19]</sup> 进化搜索<sup>[20,21]</sup> 以及其他方法<sup>[22,23,24]</sup>。在文献[4]中可以找到更多的参考文献。最近十年来,这一领域的研究趋向是不再只关注 TSP 而将 0-1 规划作为一个总体框架来研究,并且把更多的实际优化问题纳入了研究的视线。由于一种算法的有效性与特定的应用关系颇密切,所以各种 EHM 必须受到广泛实际检验才能判断其效果。此外,过去的研究大多集中于线性约束的情况,非线性约束的研究刚开始<sup>[4]</sup>。

### 4.3 采用约束平面及 HC 的 EHM

现在讨论 4.1 节给出的一般 0-1 规划问题的一种系统求解算法。这就是求得满足式(4-2)、式(4-3)和式(4-4)等 3 项约束条件且使目标函数  $F(\mathbf{X})$  (式(4-1)) 达到极小的向量  $\mathbf{X}$ 。事实上,式(4-2)和式(4-3)定义了一个  $N$  维空间中的超平面或称为多面体(polyhedron)可以用  $\varphi$  表示。而式(4-4)定义的是  $N$  维空间一个单位超立方格的各个角点(corner)可以用  $\mathbf{C} = [C_1, C_2, \dots, C_N]^T$  表示 其中  $C_i = 0$  或 1。一种直接的求解思路是将  $\mathbf{X}$  与  $\mathbf{X}$  在  $\varphi$  上的投影  $\mathbf{X}_\varphi$  之间的平方欧氏距离  $\|\mathbf{X} - \mathbf{X}_\varphi\|^2$  作为罚函数,令其乘以一个很大的正系数  $\zeta$  后与目标函数  $F(\mathbf{X})$  相加 并映射为一个 HM 的能量函数  $E(\mathbf{X})$ 。通过运行 HM 求得使  $E(\mathbf{X})$  达到极小的解 当  $\zeta$  充分大时此解必然使  $\|\mathbf{X} - \mathbf{X}_\varphi\| = 0$  这意味着  $\mathbf{X} \in \varphi$ ,即前两项约束条件得到满足。然后,再进一步解决  $\mathbf{X}$  是角点的问题,以及用 SA 或 HC 使  $E(\mathbf{X})$  从局部极小逸出并收敛到全局最小的问题。下面循此思路展开讨论。

#### 4.3.1 只存在等式约束的情形

现在将式(4-2)的  $M$  个等式约束方程写成下面的矩阵形式:

$$\mathbf{A}\mathbf{X} = \mathbf{b} \quad (4-39)$$

其中

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \vdots \\ \mathbf{a}_M^T \end{bmatrix}, \quad \mathbf{b} = [b_1, b_2, \dots, b_M]^T$$

设  $A$  的各行相互线性独立 (若否, 则可以令  $M$  减小后所余各行线性独立)。式 (4-39) 定义了一个超平面  $\varphi$ ,  $N$  维空间中任意向量  $X$  在  $\varphi$  上的投影  $X_\varphi$  可用下式计算<sup>[24,25,26]</sup>

$$X_\varphi = \Phi X + S \quad (4-40)$$

其中

$$\Phi = I - A^T(AA^T)^{-1}A$$

$$S = A^T(AA^{-1})^{-1}b$$

$\Phi$  是一个  $N \times N$  维方阵,  $I$  是一个  $N \times N$  维单位阵,  $S$  是  $N$  维列向量。由于  $A$  的秩为  $M$ ,  $AA^T$  之逆存在。易于证明 若  $X$  满足式 (4-39) 的约束 则  $X = X_\varphi$ 。反之, 若  $X = X_\varphi$  则式 (4-39) 得到满足。

将能量函数写成下列形式:

$$E(X) = F(X) + \frac{1}{2}\zeta \|X - X_\varphi\|^2 \quad (4-41)$$

引用式 (4-1) 的  $F(X)$  表达式并引用式 (4-17) 的  $E(X)$  表达式 得到

$$\frac{1}{2}X^T W X + Z^T X = \frac{1}{2}X^T Q X + g^T X + \frac{1}{2}\zeta \|X - X_\varphi\|^2$$

将式 (4-40) 代入上式并且令其左右侧的二次型和线性型参量相等, 即得到

$$\left. \begin{aligned} W &= Q - \zeta(\Phi - I) \\ Z &= g - \zeta S \end{aligned} \right\} \quad (4-42)$$

这时即可按式 (4-20) 来运行并求得解。根据这一算法运行的 HM 模拟实验结果已经证明, 只要  $\zeta$  选得充分大, 网络所收敛的稳定平衡解皆能满足式 (4-2) 给出的各项等式约束。对于具有  $M$  个城市的 TSP,  $X$  的维数  $N = M^2$  等式约束个数为  $2M$  (见式 (4-34))。在采用此方案求解时 首先应将  $W$  的各主对角元素  $w_{ii}$  置为 0 且将各  $z_i$  变为  $z_i + w_{ii}$  (见 4.2.4 小节), 改善解的质量 (即从劣质局部极小中逸出) 则必须采用 SA、HC 等算法, 见下文的讨论。

### 4.3.2 等式及不等式约束都存在的情形

式 (4-3) 的  $K$  个不等式约束可以写成下列统一的展开式:

$$r_{k1}x_1 + r_{k2}x_2 + \cdots + r_{kN}x_N \leq d_k, \quad k = 1 \sim K \quad (4-43)$$

其中  $d_k$  不限定为非负或非正。这里, 将式 (4-3) 中取“ $\geq$ ”的各不等式通过移项转换成了取“ $\leq$ ”的不等式, 从而所有约束皆取“ $\leq$ ”号。

现在引进  $K$  个新变量  $y_1 \sim y_K$  和  $K$  个系数  $\rho_k, y_k \in [0, 1], \rho_k > 0, k = 1 \sim K$ 。这样, 就可以将上列  $K$  个不等式约束转换为如下  $K$  个等式约束:

$$r_{k1}x_1 + r_{k2}x_2 + \cdots + r_{kN}x_N + \rho_k y_k = d_k, \quad k = 1 \sim K \quad (4-44)$$

$y_k$  称为松弛变量 (slack variable), 虽然将不等式约束转换为等式约束的方案很多 (在 4.4 节中还要介绍另外一种方案), 但是松弛变量都需要引入。下面讨论各  $\rho_k$  应取何值。首先, 将式 (4-43) 写成下列形式:

$$\rho_k y_k = d_k - \psi_k, \quad \psi_k = \sum_{i=1}^N r_{ki} x_i \quad (4-45)$$

可以看到,  $\rho_k y_k$  的变化范围是  $0 \sim \rho_k$ 。而  $d_k - \psi_k$  的变化范围是  $d_k - \psi_{k\max} \sim d_k - \psi_{k\min}$ ,

其中

$$\psi_{k\max} = \sum_{r_{ki} > 0} r_{ki}, \quad \psi_{k\min} = \sum_{r_{ki} < 0} r_{ki}$$

为了使  $\rho_k y_k$  的变化范围与  $d_k - \psi_k$  的变化范围是有交集部分的, 则应保证  $d_k - \psi_k$  的高端应高于  $\rho_k y_k$  的低端, 即下列条件成立:

$$d_k - \psi_{k\min} = d_k - \sum_{r_{ki} < 0} r_{ki} > 0, \quad \forall k \quad (4-46)$$

如果此式不满足, 则找不到任何一个  $\mathbf{X}$  能够满足式 (4-43) 的不等式约束, 即所提出的不等式约束条件不合理。若满足, 则  $\rho_k$  可选为大于或等于  $d_k - \psi_{k\min}$  的任何数, 一种最简单的方案是选“等于”的情况 即令

$$\rho_k = d_k - \sum_{r_{ki} < 0} r_{ki}, \quad \forall k \quad (4-47)$$

在将  $K$  个不等式约束转换成  $K$  个等式约束后, 则共有  $M+K$  个等式约束 其中前  $M$  个由式 (4-2) (或式 (4-39)) 给出, 后  $K$  个由式 (4-44) 给出。所涉及的变量共有  $N+K$  个, 即是  $x_1 \sim x_N$  以及  $y_1 \sim y_K$ , 它们构成一个如下  $N+K$  维列向量  $\hat{\mathbf{X}}$  为简便起见 记  $\hat{\mathbf{X}}$  的维数为  $N = N+K$ )

$$\mathbf{X} = [x_1, x_2, \dots, x_N, y_1, y_2, \dots, y_K]^T \quad (4-48)$$

这时式 (4-1) 的目标函数  $F(\mathbf{X})$  可以改用  $\hat{\mathbf{X}}$  来表示为  $F(\hat{\mathbf{X}})$  其计算公式是

$$F(\hat{\mathbf{X}}) = \frac{1}{2} \hat{\mathbf{X}}^T \hat{\mathbf{Q}} \hat{\mathbf{X}} + \hat{\mathbf{g}}^T \hat{\mathbf{X}} \quad (4-49)$$

其中  $\hat{\mathbf{Q}}$  是一个  $\hat{N} \times \hat{N}$  维矩阵,  $\hat{\mathbf{g}}$  是  $\hat{N}$  维列向量,

$$\hat{\mathbf{Q}} = \begin{bmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \hat{\mathbf{g}} = \begin{bmatrix} \mathbf{g} \\ \mathbf{0} \end{bmatrix} \quad (4-50)$$

$M+K$  个等式约束则可以用下列矩阵形式表示:

$$\mathbf{A}\mathbf{X} = \mathbf{b} \quad (4-51)$$

其中  $\mathbf{A}$  是  $M \times N$  维矩阵,  $\mathbf{b}$  是  $M$  维列向量,

$$\hat{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{r} & \boldsymbol{\rho} \end{bmatrix}, \quad \hat{\mathbf{b}} = \begin{bmatrix} \mathbf{b} \\ \mathbf{d} \end{bmatrix} \quad (4-52)$$

$$\mathbf{r} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1N} \\ r_{21} & r_{22} & \cdots & r_{2N} \\ \cdots & \cdots & & \cdots \\ r_{K1} & r_{K2} & \cdots & r_{KN} \end{bmatrix}, \quad \boldsymbol{\rho} = \begin{bmatrix} \rho_1 & & 0 \\ & \rho_2 & \\ & & \ddots \\ 0 & & & \rho_K \end{bmatrix}, \quad \mathbf{d} = [d_1, d_2, \dots, d_K]^T$$

注意  $\mathbf{A}$  是一个  $M \times N$  维矩阵,  $\mathbf{r}$  是一个  $K \times N$  维矩阵,  $\boldsymbol{\rho}$  是  $K \times K$  维对角阵,  $\mathbf{0}$  是  $M \times K$  维零矩阵。  $\hat{M} = M+K$ 。这样, 就可以用 4.3.1 小节所述的方法, 用具有  $\hat{N}$  个神经元的 HM 来求解此具有  $M$  个等式约束的 0-1 规划问题。下面讨论求解算法的细节以及如何提高解的质量。需要指出的是,  $\mathbf{X}$  是一个  $M = M+K$  维扩张向量, 其中前  $M$  个分量应满足  $x_i \in \{0, 1\}$  条件, 而后  $K$  个分量应满足  $y_i \in [0, 1]$ 。此二者是不同的, 这就是说  $x_i$  只能取 0 或 1 值而  $y_i$  可能取 0 至 1 之间的任何值, 这时上一小节的将各  $w_{ii}$  置为 0 的方法不能采用, 从而增加了  $x_i \in \{0, 1\}$  这一约束得到满足的难度。

### 4.3.3 采用约束平面及 HC 的 EHM 计算机模拟算法

上面两小节给出了用约束平面和松弛变量的方法，使得用 HM 解决 0-1 规划问题时式(4-2)和式(4-3)两项约束条件自然得到满足。但是还存在解的质量不高（解为低质局部极小）以及式(4-4)的约束  $x_i \in \{0,1\}$  不能保证这两项问题。此外，用数字计算机模拟这种 EHM 算法时，还存在迭代计算的步幅选择问题。现在先讨论最后这个问题（这里只讨论具有等式约束的情况。对于还具有不等式约束的情况可以用 4.3.2 小节所述方法推而广之，其中的一些特殊问题容后讨论）。

为了使计算简化，式(4-7)的 Sigmoid 函数可以用下列分段线性函数替代：

$$x_i = f_i(u_i) = \begin{cases} 1, & u_i > 1 \\ u_i, & 0 \leq u_i \leq 1 \\ 0, & u_i < 0 \end{cases} \quad (4-53)$$

其图形如图 4-4(a) 示或用图 4-4(b) 所示函数替代。而式(4-20)可以写成

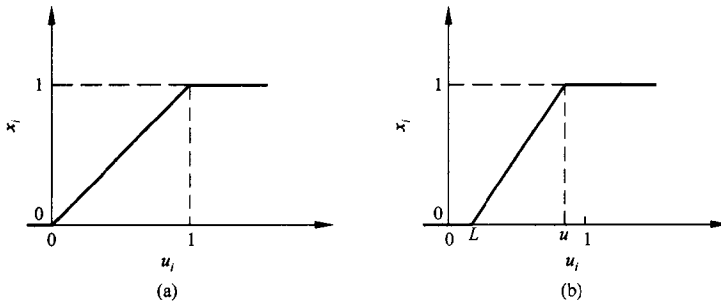


图 4-4 分段线性函数  $x_i$

下列展开形式：

$$\frac{du_i}{dt} = -W_i X - z_i, \quad i = 1 \sim N \quad (4-54)$$

其中  $W_i = [w_{i1}, w_{i2}, \dots, w_{iN}]$  是矩阵  $W$  第  $i$  行构成的行向量（见式(4-27)）。这样即可按迭代节拍  $t = 0, 1, 2, \dots$  以步幅  $\alpha$  进行下列迭代运算：

$$u_i(t+1) = u_i(t) + \alpha \left( \frac{du_i}{dt} \right)_t = u_i(t) - \alpha (W_i X(t) + z_i), \quad i = 1 \sim N \quad (4-55)$$

$$x_i(t+1) = f_i(u_i(t+1)), \quad i = 1 \sim N$$

迭代运算的初值是：

$$u_i(0)$$

和

$$x_i(0) = f_i(u_i(0)), \quad i = 1 \sim N \quad (4-56)$$

由于式(4-41)中的正系数  $\zeta$  取很大值，上列迭代运算会产生如下问题：当  $X$  在约束平面  $\varphi$  中时，式(4-41)右侧第 2 项为 0；当  $X$  脱离  $\varphi$  时，该项将猛然增大，如果步幅  $\alpha$  不是选得足够小，则由于  $du_i/dt$  非常大而使迭代算法产生震荡而不能收敛。但是， $\alpha$  太小又使收敛速度

慢得无法忍受 从而产生矛盾。此外 在采用 HC 算法时步幅  $\alpha$  应该取为随  $t$  而改变的函数  $\alpha(t)$ ，由于上述困难，这不可能实现。为此可以采取下面的方法来修正这一算法。

首先，为了避免罚函数的引入造成的  $\alpha$  难于选择的困境，将式(4-41)中  $F(\mathbf{X})$  的最小化和使  $\|\mathbf{X} - \mathbf{X}_\varphi\|$  趋于 0 这两部分功能分开施行。其次，可以看到式(4-53)和式(4-55)中的函数变换  $f_i(\cdot)$  所实现的功能是限幅功能 即在  $[0,1]$  区间中  $x_i$  与  $u_i$  完全相同 而在此区间外  $x_i$  被限制为 0 或 1。这样 可以用  $x_i$  来表示此二者并且将式(4-55)的运算分为对  $x_i$  的迭代计算和对迭代结果进行限幅两个部分，而且将后者移出来与用  $\mathbf{X}_\varphi$  替代  $\mathbf{X}$  的操作交替进行，从而使  $\mathbf{X}$  限制于  $\varphi$  且各  $x_i \in \{0,1\}$  的要求得到满足。这就是说 在每一迭代节拍  $t$ ，依次完成一个迭代运算和一个小循环运算，具体如下<sup>[19,26]</sup>

### (1) 迭代运算

只针对目标函数  $F(\mathbf{X})$  而不涉及约束条件。这时式(4-41)中的  $E(\mathbf{X}) = F(\mathbf{X})$  且式(4-42)中的  $\mathbf{W} = \mathbf{Q}$  且  $\mathbf{Z} = \mathbf{g}$ 。这时式(4-55)的第一部分运算可以表示为

$$x_i(t+1) = x_i(t) - \alpha(t) \{ \mathbf{Q}_i \mathbf{X}(t) + g_i \} \quad i = 1 \sim N \quad (4-57)$$

注意 其中已用  $x_i$  替代  $u_i$  用  $\alpha(t)$  替代  $\alpha$ 。  $\mathbf{Q}_i$  是矩阵  $\mathbf{Q}$  的第  $i$  行形成的行向量。

### (2) 小循环运算

设式(4-57)的运算结果是  $\mathbf{X}(t+1)$  为简化暂时免去时间变量 只用  $\mathbf{X}$  表示之 它就是小循环运算的输入。由于式(4-57)的迭代计算并未考虑约束的满足，所以运算所得的  $\mathbf{X}$  是脱离约束平面  $\varphi$  的。小循环运算的流程图如图 4-5 所示。

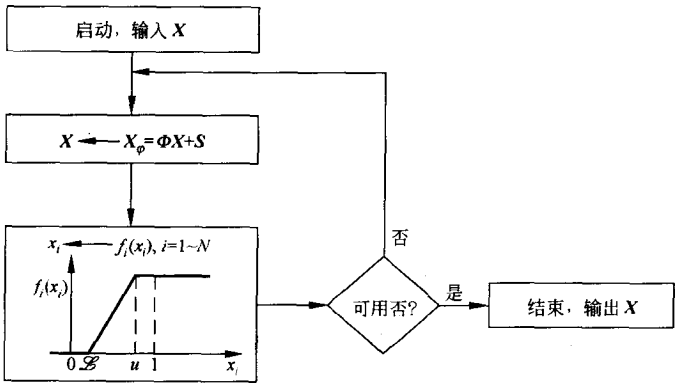


图 4-5 小循环运算流程图

第 1 步运算是用  $\mathbf{X}$  在  $\varphi$  的投影  $\mathbf{X}_\varphi$  替代  $\mathbf{X}$  见式(4-40))。第 2 步运算是将各  $f_i(x_i)$  替代  $x_i$  见式(4-53)和图 4-4(b))。第 3 步 检验前两步运算所得的  $\mathbf{X}$  是否可用。若  $\mathbf{X}$  与其在  $\varphi$  的投影  $\mathbf{X}_\varphi$  之间的欧氏距离的平方低于某预先设定的阈值，则判断  $\mathbf{X}$  为可用。这时结束循环 并将此  $\mathbf{X}$  作为下一迭代节拍  $t+1$  时的初值。若  $\mathbf{X}$  不可用，则需再度进行循环运算。下面列出基于此框架的具体运算程序。为了纳入 HC 或 SA 算法，式(4-57)中的  $\alpha(t)$  按下述规律变化：

$$\alpha(t) = \alpha_0 \cdot \text{random}[k(t), 1] \quad (4-58)$$

其中

$$k(t) = 1 - 2e^{-}$$

random[·] 表示在  $k(t)$  和 1 之间以相等的概率随机选择。这样, 当  $t = 0$  时  $\alpha(t)$  取  $\alpha_0$  或  $-\alpha_0$  的概率都是  $1/2$ ; 当  $t$  增加时  $\alpha(t)$  取  $\alpha_0$  的概率保持为  $1/2$  而另外  $1/2$  概率所取之值却从  $-\alpha_0$  逐渐增加并趋向于  $\alpha_0$ 。当  $t \rightarrow \infty$  时  $\alpha(t)$  取  $\alpha_0$  的概率为 1。此外, 对式 (4-53) 修改如下 (见图 4-4(b)):

$$x_i = f_i(u_i) = \begin{cases} 1, & u_i > 1 - u \\ u_i, & \mathcal{L} \leq u_i \leq 1 - u \\ 0, & u_i < \mathcal{L} \end{cases} \quad (4-59)$$

其中  $u$  和  $\mathcal{L}$  随迭代节拍  $t$  而变化, 在开始时 ( $t = 0$ ),  $\mathcal{L} = 0, u = 1$ ; 然后,  $t$  每增加 1,  $\mathcal{L}(t+1) = \mathcal{L}(t) + \epsilon, u(t+1) = u(t) - \epsilon$ ,  $\epsilon$  是一个小正数。这样, 随着  $t$  的增加,  $f_i(\cdot)$  的过渡区逐渐变窄, 直至成为一个阶跃式的函数。

EHM 算法程序 (采用约束面  $\varphi$  和 HC)<sup>[19]</sup> 举实例如下。

(1) 给定目标函数和约束条件参数  $Q, g, A, b$ , 根据式 (4-40) 求得约束平面  $\varphi$  的参数  $\Phi$  和  $S$ 。令  $\epsilon = 10^{-5}, \tau = 4 \times 10^5, \alpha_0 = 10^{-4}$ 。给定  $x_i$  初值为  $x_i(0) = 0.5 + \delta_i, i = 1 \sim N, \delta_i$  为在  $[-0.01, 0.01]$  范围内均匀分布的随机数。

令  $\mathcal{L}(0) = 0, u(0) = 1$ 。

(2) 令  $t = 0$ 。

(3) 计算  $k(t) = 1 - 2e^{-t/\tau}, \alpha(t) = \alpha_0 \cdot \text{random}[k(t), 1]$ 。

(4) 作迭代计算 (用  $x_i$  表示  $x_i(t)$ , “ $\leftarrow$ ” 表示用其右侧项替代其左侧项, 参见式 (4-57))

$$x_i \leftarrow x_i - \alpha(t) \{Q_i X + g_i\}, \quad i = 1 \sim N \quad (4-60)$$

(5) 作小循环运算, 其中  $f_i(x_i)$  的参数为  $\mathcal{L}(t), u(t)$ )

$$\textcircled{1} X \leftarrow X_\varphi = \Phi X + S; \quad (4-61)$$

$$x_i \leftarrow f_i(x_i), i = 1 \sim N; \quad (4-62)$$

$X$  是否可用? 若否则转入 (5); 若是则转入 (6);

(6) 计算  $\mathcal{L}(t+1) = \mathcal{L}(t) + \epsilon, u(t+1) = u(t) - \epsilon$ ;

(7) 令  $t \leftarrow t+1$  转入 (3)。

上面程序中没有给出终止条件。当  $k(t)$  与 1 足够接近或各个  $Q_i X + g_i$  都足够小时, 即可结束计算。

#### 4.3.4 应用及模拟实验举例

0-1 规划的应用很多, 其中有些是标准问题 (benchmark) 如 TSP (背包 (knapsack) 问题, 集合覆盖 (set covering) 问题等。更多的是各式各样的实际优化问题, 如车辆排序问题 (car sequencing problem, CSP), 邮递网络问题 (postal delivery network, PDN) 信道分配问题 (channel assignment problem, CAP) 货流问题 (hitchcock problem), A/D 变换器等。这一小节先介绍 CSP, PDN 和 CAP 这三个实用问题的求解。文献 [27]、文献 [28] 给出了其他一些应用实例。下一小节将讨论一些标准问题的模拟实验结果、存在问题及其对策。

## 1. CSP

设有一条汽车装配线，有  $M$  种车型需装配，记为  $m = 1 \sim M$ 。因为不同车型所需的装配时间等要求不同，同一车型不宜于紧邻在一起装配，而应隔开适当的距离，每种车型相隔的间距取决于装配该型车所耗工时及现有劳动力的多少。设装配线上共可安置  $N$  辆车，其中每种车型所占的份额是给定的。现在的问题是如何在装配线的  $N$  个位置上安排  $M$  种不同车型；在上述约束满足的条件下使得车型间隔的要求尽量得到满足。这就是 CSP。在运筹学界已有若干近似方法来解决这个问题<sup>[30,31]</sup>，下面讨论如何将 EHM 用于 CSP。

首先给出 CSP 的形式化表述。设装配线的  $N$  个位置用  $i = 1 \sim N$  表示，用下列双下标变量  $x_{mi}$  表示  $m$  车型与第  $i$  装配位的关系。

$$x_{mi} = \begin{cases} 1, & \text{第 } i \text{ 装配位是 } m \text{ 车型} \\ 0, & \text{其他} \end{cases}$$

设装配线上每种车型所占份额为  $D_m, m = 1 \sim M$ 。则 CSP 的约束条件含以下 3 项：

$$\begin{aligned} \sum_{m=1}^M x_{mi} &= 1, \quad \forall i = 1 \sim N \\ \sum_{i=1}^N x_{mi} &= D_m, \quad \forall m = 1 \sim M \\ \sum_{m=1}^M \sum_{i=1}^N x_{mi} &= \sum_{m=1}^M D_m = N \end{aligned}$$

为了描述同一车型相隔间距的要求，可以采用下列的惩罚阵列  $\mathbf{P}$ 。 $\mathbf{P}$  含  $M$  行， $N-1$  列，其第  $m$  行  $l$  列元素  $p_{ml}$  表示  $m$  车型在装配线上的序位相差为  $l$  时所受的惩罚。设装配线的第  $i$  和  $j$  位为  $m$  型车，则相应的惩罚值为  $p_{ml} = p_{m, |i-j|}$ 。表 4-2 给出了  $M=4, N=9$  的情况下一个惩罚阵列的示例。

表 4-2  $M=4, N=9$  时的一个惩罚阵列

$m \backslash l$	1	2	3	4	5	6	7	8
1	4	2	0	0	0	0	0	0
2	9	7	5	4	3	0	0	0
3	22	20	18	16	14	12	10	8
4	7	5	3	0	0	0	0	0

可以看到上表阵列中每行的非 0 元素个数  $\xi_m$  依次为  $\xi_1 = 2, \xi_2 = 5, \xi_3 = 8, \xi_4 = 3$ 。对于任何车型  $m$  当其序位差  $l = |i-j| > \xi_m$  时就不再受惩。 $\xi_m$  越大则表明  $m$  型车需更多的装配时间 因而间距更大。表 4-2 中  $m=3$  型车的  $\xi_3 = 8$  它对间距的要求最苛刻。现在，如果将  $M \times N$  个变量  $x_{mi}$  排成一个  $M \times N$  维列向量  $\mathbf{X}$ ，则可以构成下列目标函数：

$$F(\mathbf{X}) = \sum_{m=1}^M \sum_{i=1}^N \sum_{j=1}^N p_{m, |i-j|} x_{mi} x_{mj}$$

当  $i \neq j$  时 此式中的  $p_{m, |i-j|}$  已由  $\mathbf{P}$  定义。当  $i = j$  时 定义  $p_{m,0} = 0$ 。这样 CSP 归结为在前述 3 项约束条件再加上约束  $x_{mi} \in \{0,1\}, \forall m, i$  共 4 项约束条件下求  $\mathbf{X}$  使  $F(\mathbf{X})$  达到



极小。与此问题类同的尚有航线入口分配问题<sup>[32]</sup>等。

文献[19]采用4.3.3小节给出的计算程序,对各种情况的CSP进行了模拟实验并且与其他优化算法进行了比较。在所有实验中皆取 $M=4$ , $N$ 取20,40,60,80四种不同情况。惩罚阵 $P$ 用相应的 $\xi_1 \sim \xi_4$ 表示,共有五种不同类别,每种类别有其相应的 $D_1 \sim D_4$ 值(用 $D_m/N$ 来表示),表4-3给出了这五种类别的概貌。这样,4种 $N$ 值和5种类别共构成20组实验。模拟实验中除了采用具有HC和约束平面的EHM外,还采用了无HC的EHM,不依赖于HM的SA算法,SOFM算法以及一种标准的启发式算法作为对比。下面介绍模拟实验的主要结论,详情可参阅文献[33]。

表 4-3 惩罚阵  $P$  的五种类别

类别	$\xi_m, m=1 \sim 4$	$D_m/N, m=1 \sim 4$
1	(2,5,8,3)	(0.2,0.3,0.2,0.3)
2	(4,4,4,4)	(0.1,0.3,0.2,0.4)
3	(2,2,6,3)	(0.4,0.1,0.1,0.4)
4	(1,7,5,2)	(0.3,0.2,0.1,0.4)
5	(2,3,3,4)	(0.1,0.3,0.5,0.1)

(1) HC-EHM 和 SA 两种算法优于其他算法(共有5种算法参加对比),无论是平均值还是最佳值皆由这两种算法摘取金牌。

(2) 当问题的规模 $N$ 值)增大时,HC-EHM 算法所获得的解无论从平均意义还是最佳意义而言均超出了 SA 算法。

(3) HC-EHM 所得的解都是可用解。

综合而言对CSP这个特定问题的模拟实验证实了HC-EHM算法的可行、有效和实用价值。当然对于SOFM算法还应进一步研究,不应该在初步实验中效果不甚佳而立即做出否定的结论<sup>[19]</sup>。

## 2. PDN

PDN是 $p$ -中心( $p$ -hub)位置安排问题<sup>[34]</sup>的一个特例。设有 $N$ 个邮区,每个邮区有一个邮局,记为 $i=1 \sim N$ ,其位置可以用平面图上标出的一个点的坐标来表示。现在要从这 $N$ 个邮局中选出 $P$ 个( $P < N$ )作为中心局(sorting center),每个邮局都属于某个中心局。一个邮件从 $i$ 局发往 $j$ 局的过程是,首先由 $i$ 所属中心局搜集此邮件,再将其传送到 $j$ 所属中心局,最后由该中心局将其分配给 $j$ 。如果任何两个邮局 $a$ 和 $b$ (包括中心局)之间的距离用 $d_{ab}$ 表示,那么一个邮件从 $i$ 局发往 $j$ 局,且 $i$ 属中心局 $k$ 且 $j$ 属中心局 $l$ 时所需付出的代价是

$$\gamma_c D_{ik} + \gamma_t D_{kl} + \gamma_d D_{lj}$$

其中 $\gamma_c$ 是搜集一邮件时每公里所付代价, $\gamma_t$ 和 $\gamma_d$ 则是传送和分配一邮件时每公里所付代价。各局之间距离皆以公里计。现在可以定义各中心局及其他邮局对其隶属关系,用双下标变量 $x_{ik}$ 表示如下:

$$x_{kk} = \begin{cases} 1, & k \text{ 是中心局} \\ 0, & \text{其他}, k=1-N \end{cases}$$

$$x_{ik} = \begin{cases} 1, & i \text{ 属于编号为 } k \text{ 的中心局} \\ 0, & \text{其他}, i,k=1-N \end{cases}$$

由  $N^2$  个  $x_{ik}, i, k = 1 \sim N$  可以构成一个  $N^2$  维列向量  $\mathbf{X}$ 。针对此  $\mathbf{X}$  可以形成下列 PDN 的约束条件和目标函数。

约束条件：

- (1)  $\sum_{k=1}^N x_{ik} = 1, \quad \forall i = 1 \sim N$  (任何  $i$  只能属于一个中心)
- (2)  $\sum_{k=1}^N x_{kk} = P, \quad ( \text{总共设 } P \text{ 个中心}, P < N )$
- (3)  $x_{ik} \leq x_{kk}, \quad \forall i = 1 \sim N, \quad k = 1 \sim N$
- (4)  $x_{ik} \in \{0, 1\}, \quad \forall i, k = 1 \sim N$

目标函数：

$$F(\mathbf{X}) = \gamma_i \left\{ \sum_{i=1}^N \sum_{j=1}^N V_{ij} \sum_{k=1}^N \sum_{l=1}^N x_{ik} x_{jl} D_{kl} \right\} + \sum_{i=1}^N \{ \gamma_c C_i + \gamma_d D_i \} \left\{ \sum_{k=1}^N x_{ik} D_{ik} \right\}$$

其中  $V_{ij}$  是  $i$  局发往  $j$  局的邮件量 (以日计或以月计)。 $C_i$  是由  $i$  局搜集后发出去的邮件量，

$C_i = \sum_{j=1}^N V_{ij}$ 。 $D_i$  是由其他邮局发给  $i$  局的邮件量， $D_i = \sum_{j=1}^N V_{ji}$ 。易于将此目标函数写成式 (4-1) 的标准形式，

$$F(\mathbf{X}) = \frac{1}{2} \mathbf{X}^T \mathbf{Q} \mathbf{X} + \mathbf{g}^T \mathbf{X}$$

从二者的对比中，可以求得  $N^2 \times N^2$  维矩阵  $\mathbf{Q}$  和  $N^2$  维列向量  $\mathbf{g}$  的各个分量。应注意  $\mathbf{Q}$  是一个非对称的矩阵，因为  $V_{ij}$  与  $V_{ji}$  并不一定相等。

约束条件 1) 和 2) 共含  $N+1$  个等式约束，可以写成式 (4-39) 的标准矩阵形式，

$$\mathbf{A} \mathbf{X} = \mathbf{b}$$

其中  $\mathbf{A}$  是  $(N+1) \times N^2$  维矩阵， $\mathbf{b}$  是  $N+1$  维列向量。它们的各个元素可以由对比求得。约束条件 3) 是  $N^2$  个不等式约束，所以要引进  $N^2$  个松弛变量从而构成  $2N^2$  维扩张向量  $\hat{\mathbf{X}}$  (见式 4-48)) 这时目标函数将是 (见式 4-49)、式 (4-50))

$$F(\hat{\mathbf{X}}) = \frac{1}{2} \hat{\mathbf{X}}^T \hat{\mathbf{Q}} \hat{\mathbf{X}} + \hat{\mathbf{g}}^T \hat{\mathbf{X}}$$

约束条件 1)~(3) 共形成  $N+1+N^2$  个等式约束，并可写成下列矩阵形式 (见式 4-51)、式 (4-52))：

$$\hat{\mathbf{A}} \hat{\mathbf{X}} = \hat{\mathbf{b}}$$

$\hat{\mathbf{A}}$  是  $(N+1+N^2) \times 2N^2$  维矩阵， $\hat{\mathbf{b}}$  是  $N+1+N^2$  维列向量。它们的元素可在对比中求得。

文献 [19] 给出了用 EHM 求解 PDN 的模拟实验结果。作为对比，还用 SA、SOFM 和标准启发式算法等 3 种算法来解决同样的问题。实验中取  $N = 10, 15, 20$  取  $P = 2, 3$  城市分布取 A、B 两种情况，邮件传送量分 U、V 两种情况，这样可以组合为 24 种模拟实验。在所有实验中皆取  $\gamma_c = 3.2, \gamma_d = 0.5, \gamma_i = 0.4$ 。实验中所用的程序按 4.3.3 小节所给出的算法。只是对于 PDN 而言采用 HC 和不用 HC 所得效果差别不大，所以采用了无 HC 的 EHM，即  $\alpha(t) \equiv \alpha_0$  而不是  $\alpha_0 \cdot \text{random}[k(t), 1]$ 。其他方面则严格按该程序执行。24 项模拟实验的结果是，EHM 解与最佳解的相对误差平均值为 0.008 而其他 3 种方法则在

0.8% ~ 4.3% 之间。

### 3. CAP

随着数字蜂窝移动通信系统的高速发展,如何更有效地利用电磁频谱资源就成了极重要的问题。这需要将有限个数的频道分配给各个蜂窝小区 (cell,region) 在通信业务得到满足的条件下使平均信号干扰比 (signal to interference ratio,SIR) 达到最大。当各小区的业务量固定时可以采取静态的信道分配方案,相应的问题称为 SCA(static channel assignment)。如果业务量不断变化,则可用动态信道分配,即 DCA(dynamic channel assignment)。后者较困难并且以前者为基础。下面只讨论针对 SCA 的 CAP 所以就采用 SCA 这个标记。SCA 是 NP 难题 用 SA 式算法<sup>[35]</sup> 和其他人工神经网络方法<sup>[36,37]</sup> 来解决此问题一直受到重视。此处介绍用 EHM 解决此问题<sup>[7]</sup>

设有  $N$  个蜂窝小区 记为  $i = 1 \sim N$  再设有  $M$  个频率依次相邻的信道,记为  $m = 1 \sim M$ 。为了描述这  $M$  个信道如何分配给  $N$  个小区,可以用双下标变量  $x_{mi}$  其定义为:  $x_{mi} = 1$ , 当  $m$  信道分配给  $i$  小区,  $x_{mi} = 0$  当  $m$  信道未分配给  $i$  小区。由于一个频道可以同时分配给几个小区,  $x_{mi} = 1$  不意味  $x_{mj} = 0, j \neq i$ 。这样用  $N \times M$  个变量  $x_{mi}$  可以构成一个  $MN$  维列向量  $\mathbf{X}$  在一定约束条件下每一种  $\mathbf{X}$  代表了一种信道分配方案。SCA 就是在此约束下求一最佳  $\mathbf{X}$  使得平均 SIR 达到最大或者使得相应的目标函数  $F(\mathbf{X})$  达到极小。下面即给出此课题的形式化表述。

SCA 约束:

$$(1) \quad \sum_{m=1}^M x_{mi} = D_i, \quad i = 1 \sim N$$

其中  $D_i$  即为根据业务量确定的第  $i$  小区所需的信道个数。注意在 SCA 中  $D_i$  是预先给定的常数。

$$(2) \quad x_{mi} \in \{0,1\} \quad \forall m = 1 \sim M, i = 1 \sim N$$

SCA 目标函数:

为了给出目标函数,首先需确定同一小区及不同小区使用的信道间隔不同时所造成的干扰,显然信道间隔越小干扰越大。为简化起见,采用下面的近似计算方法。设  $i$  小区用  $m$  信道且  $j$  小区用  $n$  信道 若由此造成的干扰用  $p_{i,j,|m-n|}$  表示。

当  $i \neq j$  时,设  $p_{i,j,0} = C_{ij}$ ,  $C_{ij}$  为  $i, j$  两个小区使用同一频道时造成的干扰,这显然是一种干扰最大的情况。 $C_{ij}$  是一个正整数,其值取决于  $i, j$  两个小区的相距遥远程度,若很近则  $C_{ij}$  大 反之则小。当  $|m - n| = l \neq 0$  时,  $p_{i,j,l}$  可用下式计算:

$$p_{i,j,l} = \begin{cases} p_{i,j,l-1} - 1, & l = 1 \sim C_{ij} - 1 \\ 0, & l \geq C_{ij} \end{cases}$$

$C_{ij}$  在模拟实验中须预先给定,在实际环境中则是由测试决定的。例如  $C_{13} = 3$  表明 1,3 两个小区共用一个频道时干扰量为 3,当使用频道间隔扩大时干扰量渐降。当  $|m - n| \geq 3$  时不再存在干扰。

当  $i = j$  时,  $p_{i,i,0} = 0$ , 这是因为在同一小区中使用同一频道意味着将其分配给一个用户而不可能分配给两个用户,所以不造成干扰。当  $|m - n| = l \neq 0$  时,可用下式计算  $p_{i,i,l}$ :

$$p_{i,i,l} = \begin{cases} C_{ii}, & l = 1 \\ p_{i,i,l-1} - 1, & l = 2 \sim C_{ii} \\ 0, & l > C_{ii} \end{cases}$$

$C_{ii}$  是一个正整数,表示同一区内使用相邻信道时造成的干扰,例如  $C_{11} = 2$ 。一般情况下对于不同的小区  $i$ ,  $C_{ii}$  是相同的,例如可取  $C_{ii} = 2, \forall i = 1 \sim N$ 。这样,可以构成如下的目标函数:

$$F(\mathbf{X}) = \sum_{i=1}^N \sum_{j=1}^N \sum_{m=1}^M \sum_{n=1}^M p_{i,j,|m-n|} x_{mi} x_{nj}$$

当  $F(\mathbf{X})$  达到极小时干扰量最小,从而使 SIR 达到最大。这样,SCA 的任务即在满足 (1)、(2) 两项约束的条件下求  $\mathbf{X}$  使  $F(\mathbf{X})$  达到极小。由于  $F(\mathbf{X})$  中只有二次项而没有线性项,对比式 (4-1),可以将其写成  $F(\mathbf{X}) = \frac{1}{2} \mathbf{X}^T \mathbf{Q} \mathbf{X}$ ,其中  $\mathbf{X}$  是  $NM$  维列向量,  $\mathbf{Q}$  是  $NM \times NM$  维矩阵,其元素可由对比求得。值得注意的是  $\mathbf{Q}$  的对角元素皆为 0,因此是一个非定 (indefinite) 矩阵。这时  $F(\mathbf{X})$  的局部极小的个数很多。SCA 的约束条件 (1) 可以写成矩阵形式  $\mathbf{A}\mathbf{X} = \mathbf{b}$  其中  $\mathbf{A}$  是  $N \times NM$  维矩阵,  $\mathbf{b} = [D_1, D_2, \dots, D_N]^T$ 。

文献 [7] 给出了按 4.3.3 小节给出的 HC-EHM 计算程序求解 SCA 时的模拟实验结果,作为对比,还选择了另外两种启发式算法,SA 算法、SOFM 算法和无 HC 的 EHM 等 5 种算法来解同样的问题。模拟实验包括  $N = 21, M = 21, 37, 56, 91$  所形成的 4 种情况,其约束条件 (1) 中的  $D_1 \sim D_{21}$  为 (2, 6, 2, 2, 2, 4, 4, 13, 19, 7, 4, 4, 7, 4, 9, 14, 7, 2, 2, 4, 2) 或为 (1, 1, 1, 2, 3, 6, 7, 6, 10, 10, 11, 5, 7, 6, 4, 4, 7, 5, 5, 5, 6) 其  $C_{ii} = 2$  或 3,  $C_{ij}$  则取决于六角形的蜂窝小区位置安排。在模拟实验中 21 个蜂窝小区的位置如图 4-6 所示。对于任何一

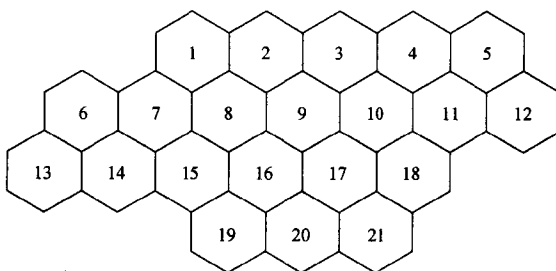


图 4-6 蜂窝小区的位置图

个小区只有环绕它的第一圈和第二圈会对其产生干扰;第一圈中的  $C_{ij} = 2$  第二圈中的  $C_{ij} = 1$ 。例如,对于  $i = 3$  的小区,  $C_{32} = C_{39} = C_{3,10} = C_{34} = 2, C_{31} = C_{38} = \dots = C_{3,11} = C_{3,5} = 1, C_{37} = C_{3,15} = \dots = C_{3,20} = C_{3,21} = C_{36} = 0$ 。模拟实验还包括了赫尔辛基市  $24\text{km} \times 21\text{km}$  范围内根据实际地图构成的 SCA 问题<sup>[38]</sup>。其中  $N = 25, M = 73, \mathbf{b}^T = [D_1, D_2, \dots, D_{25}]^T = [10, 11, 9, 5, 9, 4, 5, 7, 4, 8, 8, 9, 10, 7, 7, 6, 4, 5, 5, 7, 6, 4, 5, 7, 5]^T$ ,  $C_{ii} = 2, C_{ij} = 1$  或 0。另外,还在此基础上构成了 3 个缩小的赫尔辛基市 SCA 问题,它们是  $N = 10, M = 30; N = 15, M = 44; N = 20, M = 60$ 。其他参数则可由原始的赫尔辛基市 SCA 问题中获取。这样,文献 [7] 共用 6 种优化方法对这 8 种情况做了模拟实验,每个实验

中取多种初值从而找到多种解，用这些解的平均值和最佳值来比较 6 种解法的效果。结论如下。

(1) HC-EHM 所求得解都是可用解。

(2) 无论用平均标准还是最佳标准进行比较，HC-EHM 都是最优的。两种启发式算法及 SOFM 算法排在最后 3 位 其干扰值较 HC-EHM 算法大很多。SA 算法和无 HC 的 EHM 算法排在中间，在 8 项实验中 有 4 ~ 5 项质量较好，而其他质量较差。

#### 4.3.5 用 EHM 求解一些标准 0-1 规划问题的讨论

虽然 4.3.4 小节中介绍了一些 EHM 解决实际优化问题的例子，在这些例子以及其他很多例子中，EHM 确实优于其他优化算法（包括 SA 算法和其他人工神经网络算法），但是关于 EHM 的研究和争论远远没有结束。关注的焦点是用 EHM 来解决 0-1 规划问题时有不少理论问题尚未解决，尤其是用它来解一些标准问题（benchmark），如背包问题、TSP 等问题时还存在若干疑问。现在研究者的普遍共识是用约束平面的方法可以使式(4-2) ~ 式(4-4) 给出的 3 项约束条件中的 (1)、(2) 两项轻而易举地得到满足，从而不形成任何障碍，而第 (3) 项，即  $x_i \in \{0,1\}$ ，则不一定能轻易实现。另外，如何保证算法能收敛到  $E(\mathbf{X})$  的全局最小从而使目标函数  $F(\mathbf{X})$  达到全局极小也还存在疑问，问题在于当  $F(\mathbf{X})$  的局部极小点的个数非常多时，即使在算法中引入了 SA, HC 算法也不能保证收敛到全局极小，也不能保证收敛到质量好的局部极小点。特别是，有些问题（如 TSP）的目标函数  $F(\mathbf{X})$  既可以表示成二次型又可以表示成线性型（二者的约束条件当然有很大差异），后者的局部极小点较前者无疑要少得多。在运筹学界解决 TSP 时用线性型的目标函数，而 HM 则为避免约束实现的困难而采用二次型目标函数来解 TSP，这也给不同方法的比较带来困难。这一小节将以背包问题和 TSP 这两个标准问题为例 来讨论 EHM 用于组合优化时互有关联的这样一些问题：（1）解的可用性约束中  $x_i \in \{0,1\}$  条件能否满足（这涉及约束平面是整数面还是非整数面，见 4.3.5 小节的 (1)）。（2）目标函数  $F(\mathbf{X})$  是二次型还是线性型。（3）如何能获得全局最优解。如不能得到全局最优，如何能尽量改善局部极小解的质量以及如何评价 EHM 的解与其他经典优化方法所获解的优劣差异。

##### 1. 背包问题

设有  $N$  种物品 记为  $i = 1 \sim N$  其价值为  $C_i, i = 1 \sim N$ 。现在要从这  $N$  种物品中选出若干种放在一个背包里，而背包可容物品的总重量、总体积……是有限的且每种物品的重量、体积……是一定的。现在要从中选出若干种物品装入背包，在其重量、体积……不超过容许值的条件下，其总价值达到极大。为此可设  $N$  个变量  $x_i, i = 1 \sim N$  若  $i$  被选中  $x_i = 1$  反之  $x_i = 0$ 。各  $x_i$  构成向量  $\mathbf{X} = [x_1, x_2, \dots, x_N]^T$ 。这样 背包问题可定义为在满足下列两项约束的条件下求  $\mathbf{X}$  使总价值  $C(\mathbf{X})$  达极大。约束条件：

$$(1) \quad \sum_{i=1}^N a_{ki} x_i \leq b_k, \quad k = 1 \sim K \quad (4-63)$$

$$(2) \quad x_i \in \{0,1\} \quad (4-64)$$

价值函数：

$$C(\mathbf{X}) = \sum_{i=1}^N C_i x_i, \quad C_i > 0, \forall_i \quad (4-65)$$

由于 0-1 规划问题习惯于求一使目标函数  $F(\mathbf{X})$  达其最小值的解。所以用下列  $F(\mathbf{X})$  代替  $C(\mathbf{X})$ ：

$$F(\mathbf{X}) = \sum_{i=1}^N g_i x_i, \quad g_i = -C_i, i = 1 \sim N \quad (4-66)$$

这样，背包问题的形式化表述为：

在满足 (1), (2) 两项约束的条件下求  $\mathbf{X}$  使  $F(\mathbf{X}) = \min$ 。

在式 (4-63) 中不等式约束的个数  $K$  不得小于 1。 $C_i, a_{ki}, b_k$  等皆为正实数。由于  $F(\mathbf{X})$  是线性函数，它不存在局部极小，因此表面上看只要在约束平面内用最陡下降算法即可求得使  $F(\mathbf{X})$  达到  $\min$  的解。但是，问题远非这样简单。首先，它是 NP 完备的<sup>[39,40]</sup>。其次，下面将用例子说明，背包问题的约束平面  $\varphi$  为非整数面，这时约束条件 (2) (式 (4-64)) 很难满足。对于神经网络学界，用 HM 求解背包问题和求解 TSP 一样成为标准难题。在 4.4 节和 4.5 节还要探讨解此问题的其他方案。设有一个  $N = 2$  的背包问题，其目标函数  $F(\mathbf{X})$  和约束条件如下所列：

$$F(\mathbf{X}) = -x_1 - 2x_2$$

$$x_1 + x_2 \leq 1.7 \quad \text{且} \quad x_i \in \{0, 1\}, \quad i = 1, 2$$

这个问题可以直观求得解为  $x_1 = 0, x_2 = 1$ 。现在用 EHM 来求其解。这里用的是 4.3.2 小节给出的框架及 4.3.3 小节给出的实际计算机模拟算法。由于背包问题的目标函数是一个线性函数，线性函数是凸函数，所以不存在局部极小点，这样该模拟算法中的 HC 部分 (式 4-58) 不再需要，予以取消。此外，函数  $x_i = f_i(u_i)$  也不用式 (4-59) 的变过渡区形式而采用式 (4-53) 的固定过渡区形式 (其原因将在下面解释)。由于此背包问题中  $N = 2$ 、 $K = 1$ ，因此可以引入一个松弛变量  $y_1$ ，形成  $N = N + K = 3$  维列向量  $\mathbf{X} = [x_1, x_2, y_1]^T$  (见式 (4-48)) 并形成关于  $\mathbf{X}$  的具有等式约束的 0-1 规划问题，其约束平面  $\varphi$  由下式给出 (见式 4-44) (式 4-47))：

$$x_1 + x_2 + 1.7y_1 = 1.7$$

$\varphi$  是 3 维空间中的一个 2 维平面，如图 4-7 所示， $\varphi$  与单位立方体 (图 4-7 中实线所示) 有 5 个交点： $\mathbf{X}_a = [0.0, 0.0, 1.0]^T$ ， $\mathbf{X}_b = [1.0, 0.0, 0.41]^T$ ， $\mathbf{X}_c = [1.0, 0.7, 0.0]^T$ ， $\mathbf{X}_d = [0.7, 1.0, 0.0]^T$ ， $\mathbf{X}_e = [0.0, 1.0, 0.41]^T$ 。这些交点也称为  $\varphi$  的角点 (vertex)，EHM 所搜求的解即这些角点中的某一个。按照 4.3.3 小节给出的计算机程序 (注意，进行上述两项修正)，就可以求得此问题的解<sup>[26]</sup>。模拟实验所得的解为系统收敛到  $\hat{\mathbf{X}}_d$ ，这时  $x_1 = 0.7, x_2 = 1.0, y_1 = 0.0$ 。可以看到，如果放弃背包问题的约束条件 (2) (即式 4-64)，则这个解就是所能求得的最优解，这种解称为背包问题的 LPR (linear

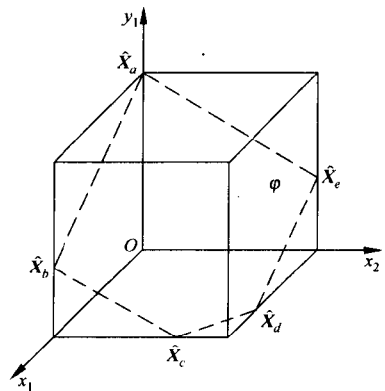


图 4-7  $\varphi$  的示意图

programming relaxation 解。但是,如果要求约束条件(2)必须满足,则此解不符合需要。这样,因为 EHM 只能收敛到这个松弛解而不可能再逸出而收敛到符合需要的解上,从而无法用于背包问题。稍作分析可以看到,前述的 5 个交点中只有 3 个符合约束条件(2)( $\hat{X}_a, \hat{X}_b, \hat{X}_c$ ),而另外两个不符合。如果所有交点都符合约束条件(2)相应的约束面  $\varphi$  称为整约束面(integral polytope)。相反,则  $\varphi$  称为非整约束面(nonintegral polytope)。对于前者, EHM 的解将收敛到这些交点中的某一个,这时约束条件(2)总能满足。对于后者,则可能收敛到不符合条件(2)的交点。上面举的例子及一般的背包问题正属于这种情况。

可以设想,对于非整约束面的情况,能否采取某种模拟退火的方法,将解从不符合条件(2)的交点中拉出来后收敛到符合条件的交点。对于上面举的例子,则希望从收敛点  $X_d$  解脱出来并收敛到所需解  $X_c$ 。一种简单的设想是在能量函数中再加上一个退火函数  $E_{SA}(X)$ :

$$E_{SA}(X) = -\gamma \sum_{i=1}^N (x_i - 0.5)^2$$

注意其中只包括各待求变量  $x_1 \sim x_N$  而不包括松弛变量  $y_1 \sim y_k$ 。如果  $\gamma > 0$  则只有当约束条件(2)  $x_i \in \{0, 1\}, \forall i$  得到满足时,  $E_{SA}(X)$  达到其极小值。而且对于任何满足此约束条件的  $X = [x_1, x_2, \dots, x_N]^T$  它所对应的  $E_{SA}(X)$  都是相同的,因此不会对合乎要求的解究竟是哪一个产生影响。具体作法是:迭代开始时,  $\gamma$  取为一个大负数,这时  $E_{SA}(X)$  具有下凸(convex)型,其最小点在  $X = [0.5, 0.5, \dots, 0.5]^T$ , 这时退火项的作用是驱使  $X$  趋向此“内点”。随着迭代进行,  $\gamma$  逐渐由负向正过渡并最终取一个大正数,这时  $E_{SA}(X)$  具有上凹(concave)型,其最小点在  $X$  的各分量  $x_i \in \{0, 1\}$  处。这时退火项的作用是趋使  $X$  趋向各符合约束条件(2)的点。模拟实验结果证明,以上设想并不成功。原因在于,  $\gamma$  最终取的大正数如果较式(4-41)约束项的系数  $\zeta$  小,则根本不能从  $X_d$  这个不可用解中逸出;反之,当  $\gamma$  与  $\zeta$  可比拟时,虽然可以从  $X_d$  逸出但是约束项将不再起约束作用,这时将收敛到  $X = [1.0, 1.0]^T$  这个解上,显然这也是不合用的。文献[26]讨论了 SA 算法失效的深层次原因。总之,对于像背包问题这样的非整约束面 0-1 规划问题,在 EHM 算法中无论结合哪一种 SA 方法(包括式(4-59)的变过渡区形式)都不能解决所求得解必定满足约束条件(2)的问题。下一节将转换思路,用 Tabu 搜索来解决此问题。然后,在 4.3.5 小节的 3 中再结合包括 TSP 在内的其他 0-1 规划问题来讨论非整约束面与目标函数  $F(X)$  选择之间的关系。

## 2. 非整约束面的 Tabu 搜索

Tabu 的原意是禁忌,Tabu 搜索的初衷是在优化问题的搜索空间中可以划出某些没有必要进行搜索(例如,其明显不符合约束或离最优解差距明显很大)的禁忌区,这样使搜索可以在其他更有可能获得成功的区域中进行,从而提高搜索效率。将 Tabu 搜索的概念移植到非整约束面 0-1 规划问题的 EHM 求解时,可以将目标函数设计成动态的,即  $F(X)$  被设计成随迭代节拍  $t$  而变化的。其思路是在  $F(X)$  中再增加一个惩罚项,

即对于  $t$  以前系统刚访问过的解进行惩罚, 离  $t$  越近的解惩罚得越重。这样, 只要系统的运作规律及参数选得合适, 那么即使系统陷入某个松弛解 (如前举例中的  $\mathbf{X}_d$ ) 时, 也会因为在进一步的运行中受到惩罚而从中解脱出来, 从而使系统有可能访问并求得约束面  $\varphi$  与单位超立方体交点中的其他解。下面给出含此惩罚项的随  $t$  而变化的目标函数  $F_t(\mathbf{X}, t)$ 。

$$F_t(\mathbf{X}, t) = F(\mathbf{X}) + \mathcal{F}_t(\mathbf{X}, t) \quad (4-67)$$

其中  $F(\hat{\mathbf{X}})$  由式 (4-49) 和式 (4-50) 决定。惩罚项  $\mathcal{F}_t(\hat{\mathbf{X}}, t)$  由下式决定:

$$\mathcal{F}_t(\hat{\mathbf{X}}, t) = \beta \int_0^t e^{\alpha(S-t)} p[\hat{\mathbf{X}}, f(\hat{\mathbf{X}}(S))] dS \quad (4-68)$$

其中  $\mathbf{X}(S)$  表示  $0 \leq S \leq t$  时系统的  $\mathbf{X}$  的值 被称为“状态” $\alpha$  和  $\beta$  是两个正数,  $f(\cdot)$  是下面将讨论的一个向量函数且满足  $f(\mathbf{X}(S)) \approx \mathbf{X}(S)$ 。函数  $p[\mathbf{A}, \mathbf{B}]$  是  $\mathbf{A}$ 、 $\mathbf{B}$  二向量相似度的度量, 二者越相似其取值越大。

现在研究当系统在某个时刻 (不失一般性设其为  $t=0$ ) 陷入了  $\varphi$  的某个角点 设其为  $\mathbf{X}_0$  时 应如何选择函数  $f(\cdot)$  以及选择参数  $\alpha$  和  $\beta$  才能使系统从陷阱中拔出来。先研究问题的前一部分 后一部分则稍后再讨论。首先 给出函数  $p(\cdot, \cdot)$  如下:

$$p(\mathbf{A}, \mathbf{B}) = -\|\mathbf{A} - \mathbf{B}\|^2 \quad (4-69)$$

可以看到  $\mathbf{A}$  与  $\mathbf{B}$  越接近时  $p$  的取值越大且其为解析函数, 便于进行微分运算。若  $t=0$  时系统陷入一个角点  $\hat{\mathbf{X}}_0$  即  $\hat{\mathbf{X}}(0) = \hat{\mathbf{X}}_0$  且当  $S \geq 0$  时皆有  $\hat{\mathbf{X}}(S) = \hat{\mathbf{X}}_0$ 。再利用式 (4-69), 则式 (4-68) 可以写成

$$\mathcal{F}_t(\hat{\mathbf{X}}, t) = -\beta \left[ \int_0^t e^{\alpha(S-t)} dS \right] \|\hat{\mathbf{X}} - f(\hat{\mathbf{X}}_0)\|^2, \quad t > 0$$

可以看到  $\mathcal{F}_t(\hat{\mathbf{X}}, t)$  是一个最大点在  $f(\hat{\mathbf{X}}_0)$  的二次函数。如果选  $f(\hat{\mathbf{X}}_0) = \hat{\mathbf{X}}_0$  则此最大点就在  $\mathbf{X}_0$ 。另外, 不包含此惩罚项的原能量函数  $E(\hat{\mathbf{X}})$  在  $\mathbf{X}_0$  处为一局部极小点, 由于  $\hat{\mathbf{X}}_0$  是一个角点  $E(\mathbf{X})$  在  $\mathbf{X}_0$  附近的变化曲线是不可微的, 如图 4-8 所示。这样, 按照上选的  $(\mathbf{X}_0)$ ,  $E(\mathbf{X})$  与  $\mathcal{F}_t(\mathbf{X}, t)$  之和在  $\mathbf{X}_0$  处将永远存在局部极小 如图 4-8(a) 所示 (为简化起见, 给出的是 2 维情况) 但是如果令  $f(\hat{\mathbf{X}}_0) = \hat{\mathbf{X}}_1 = \hat{\mathbf{X}}_0 + \epsilon (\hat{\mathbf{X}}_0 - \hat{\mathbf{X}}_2)$  其中  $\epsilon \ll 1, \hat{\mathbf{X}}_2 \in \varphi$  则  $\hat{\mathbf{X}}_1 \neq \hat{\mathbf{X}}_0$  但是二者比较接近 如图 4-8(b) 所示。这时只要  $\alpha, \beta, \epsilon$  诸参数选得合适 在  $\hat{\mathbf{X}}_0$  及其周围不再存在任何局部极小点, 这时系统的状态  $\mathbf{X}$  就可以从陷阱  $\mathbf{X}_0$  中逸出。之所以选择  $\mathbf{X}_2$  在  $\varphi$  内 是为了保持系统状态  $\mathbf{X}$  的迁移始终在约束平面  $\varphi$  之内。下面给出含有此惩罚项的 Tabu 搜索 EHM 的运行方程。

首先 利用式 (4-69) 可将式 (4-68) 写成

$$\mathcal{F}_t(\hat{\mathbf{X}}, t) = -\frac{1}{2} \theta(t) \hat{\mathbf{X}}^T \hat{\mathbf{X}} - \mathbf{h}^T(t) \hat{\mathbf{X}} + g(t) \quad (4-70)$$

其中



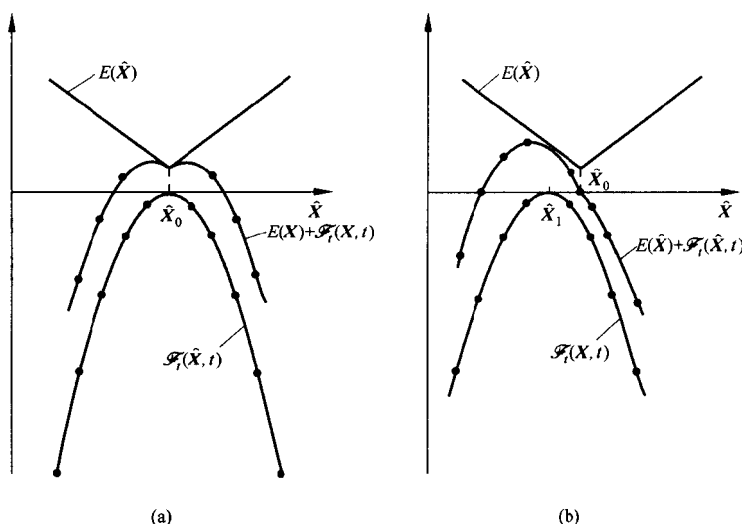


图 4-8  $E(\mathbf{X})$ 、 $F_t(\mathbf{X}, t)$  和  $E(\mathbf{X}) + F_t(\mathbf{X}, t)$  的函数示意图

$$\theta(t) = 2\beta \int_0^t e^{\alpha(S-t)} dS \quad (4-71)$$

$$\mathbf{h}(t) = -2\beta \int_0^t e^{\alpha(S-t)} \mathbf{f}(\hat{\mathbf{X}}(S)) dS \quad (4-72)$$

$$g(t) = -\beta \int_0^t e^{\alpha(S-t)} \mathbf{f}^T(\hat{\mathbf{X}}(S)) \mathbf{f}(\hat{\mathbf{X}}(S)) dS \quad (4-73)$$

式(4-70)右侧第3项  $g(t)$  与  $\mathbf{X}$  无关 所以在 EHM 的 Tabu 搜索中不起作用, 在后面的计算中可以将它去除。该式右侧第 1、2 项的系数  $\theta(t)$  和  $\mathbf{h}(t)$  的计算需用到  $\mathbf{X}(S)$ ,  $0 \leq S \leq t$  而后的计算又需要用到前者(见下文)所以只能采取交互迭代计算的方法。即对于每一个时刻  $t$  分两步由  $\mathbf{X}(t-\Delta t)$ ,  $\theta(t-\Delta t)$ ,  $\mathbf{h}(t-\Delta t)$  求  $\mathbf{X}(t)$ ,  $\theta(t)$ ,  $\mathbf{h}(t)$  (注意 在模拟计算中以  $\Delta t$  为间隔对  $t$  取离散值  $\Delta t \ll 1$ )。

首先 第一步求  $\theta(t)$  和  $\mathbf{h}(t)$ ,

$$\theta(t) = \theta(t-\Delta t) + \left. \frac{d\theta}{dt} \right|_{t-\Delta t} \cdot \Delta t, \left. \frac{d\theta}{dt} \right|_{t-\Delta t} = 2\beta - \alpha\theta(t-\Delta t) \quad (4-74)$$

$$\mathbf{h}(t) = \mathbf{h}(t-\Delta t) + \left. \frac{d\mathbf{h}}{dt} \right|_{t-\Delta t} \cdot \Delta t, \left. \frac{d\mathbf{h}}{dt} \right|_{t-\Delta t} = -2\beta \mathbf{f}(\hat{\mathbf{X}}(t-\Delta t)) - \alpha \mathbf{h}(t-\Delta t) \quad (4-75)$$

此迭代计算的初值是  $\theta(0) = 0$ ,  $\mathbf{h}(0) = \mathbf{0}$ 。第二步 固定  $\theta(t)$  和  $\mathbf{h}(t)$ , 用下述方法计算  $\mathbf{X}(t)$ 。这里 将 Tabu 搜索的目标函数  $F_t(\mathbf{X}, t)$  (或 4-67)) 代入 EHM 的能量函数  $E(\mathbf{X})$  式(4-41)) 注意式(4-41)是针对只有等式约束问题的, 此处的背包问题为非等式约束问题, 所以该式中的变量  $\mathbf{X}$  换成  $\hat{\mathbf{X}}$  并且将  $g(t)$  扣除 即得到

$$\begin{aligned}
 E(\hat{\mathbf{X}}) &= F(\hat{\mathbf{X}}) + \mathcal{F}_t(\hat{\mathbf{X}}, t) - g(t) + \frac{1}{2}\zeta \|\hat{\mathbf{X}} - \hat{\mathbf{X}}_\varphi\|^2 \\
 &= \frac{1}{2}\hat{\mathbf{X}}^T \hat{\mathbf{W}} \hat{\mathbf{X}} + \hat{\mathbf{Z}}^T \hat{\mathbf{X}}
 \end{aligned} \quad (4-76)$$

将式(4-40)、式(4-49)、式(4-70)代入上式。注意,在用式(4-40)时应该将 $\mathbf{X}, \mathbf{X}_\varphi, \mathbf{A}, \mathbf{b}, \Phi, \mathbf{S}$ 等换成 $\hat{\mathbf{X}}, \hat{\mathbf{X}}_\varphi, \hat{\mathbf{A}}, \hat{\mathbf{b}}, \hat{\Phi}, \hat{\mathbf{S}}$ (见式(4-50)、式(4-51))。还要注意式(4-49)的 $F(\hat{\mathbf{X}})$ 表达式中既含二次型又含线性项,这里将二者都包含进去以解决一般问题(对背包问题则只含线性项)即得到

$$E(\hat{\mathbf{X}}) = \frac{1}{2}\hat{\mathbf{X}}^T \hat{\mathbf{Q}} \hat{\mathbf{X}} + \hat{\mathbf{g}}^T \hat{\mathbf{X}} - \theta(t)\hat{\mathbf{X}}^T \hat{\mathbf{X}} + \mathbf{h}^T(t)\hat{\mathbf{X}} + \frac{1}{2}\zeta \|\hat{\mathbf{X}} - (\hat{\Phi}\hat{\mathbf{X}} + \hat{\mathbf{S}})\|^2 \quad (4-77)$$

对比式(4-76)和式(4-77)即可得

$$\mathbf{W} = \mathbf{Q} - \zeta(\Phi - \mathbf{I}) - \theta(t)\mathbf{I} \quad (4-78)$$

$$\mathbf{Z} = \hat{\mathbf{g}} - \zeta\mathbf{S} - \mathbf{h}(t) \quad (4-79)$$

这样,即可用4.3.3小节给出的迭代算法,由 $\hat{\mathbf{X}}(t - \Delta t)$ 及已知的 $\theta(t)$ 和 $\mathbf{h}(t)$ 求 $\hat{\mathbf{X}}(t)$ ,只是取消其中的HC和SA部分的算法,即式(4-59)中的 $u$ 和 $\mathcal{L}$ 皆取常数值( $u = 1, \mathcal{L} = 0$ )以及式(4-60)中的 $\alpha(t)$ 也取常数值 $\alpha(t) = \alpha_0 \ll 1$ 。这时式(4-60)可表示为

$$\hat{x}_i(t) \leftarrow \hat{x}_i(t - \Delta t) - \alpha_0 \{ (\hat{\mathbf{Q}} - \theta(t)\mathbf{I})_i \hat{\mathbf{X}}(t - \Delta t) + \hat{g}_i - h_i(t) \}, \quad i = 1 \sim \hat{N} \quad (4-80)$$

其中 $(\mathbf{Q} - \theta(t)\mathbf{I})_i$ 是相应矩阵第 $i$ 行形成的行向量, $\hat{g}_i$ 和 $h_i(t)$ 分别为 $\hat{\mathbf{g}}$ 和 $\mathbf{h}(t)$ 的第 $i$ 个分量。而式(4-61)和式(4-62)仍可应用(只需将其中的 $\mathbf{X}, x_i, \Phi, \mathbf{S}, N$ 换成 $\hat{\mathbf{X}}, \hat{x}_i, \hat{\Phi}, \hat{\mathbf{S}}, \hat{N}$ )。

对于背包问题,因为只含线性项,即有 $\mathbf{Q} = \mathbf{0}$ 这时 $F_t(\hat{\mathbf{X}}, t)$ 在任何时间 $t$ 都是线性函数或凸函数,不存在局部极小点。所以无须考虑从 $F_t(\hat{\mathbf{X}}, t)$ 的局部极小点中逸出的问题,而只需考虑系统陷入 $\varphi$ 的某个角点时逸出的问题。文献[26]给出的结论是:(1) $\hat{\mathbf{X}}_2$ 的选择,以 $\mathbf{X}_2 = \mathbf{S}$ 最恰当(易证 $\hat{\mathbf{S}} \in \varphi$ )。(2) $\alpha, \beta, \epsilon$ 的选择,对于最劣情况,只要按1)来选 $\mathbf{X}_2$ 且满足下式时系统可以从 $\varphi$ 的任何角点逸出:

$$\frac{\beta\epsilon}{\alpha} \geq \frac{\|\hat{\mathbf{g}}\|}{2} \quad (4-81)$$

对于4.3.5小节1开头处所举的 $N = 2$ 的背包问题示例(见图4-7)文献[26]给出的参数是 $\alpha = 0.01, \beta = 2.0, \epsilon = 0.1$ 。模拟实验结果表明系统能遍历 $\varphi$ 的5个角点 $\hat{\mathbf{X}}_a \sim \hat{\mathbf{X}}_e$ 。对于另一个维数 $N$ 较高的背包问题则选 $\alpha = 0.01, \beta = 1.0, \epsilon = 0.1$ 也得到了从不符合约束条件(2)的角点中逸出的目标,并且所得到的解与目前最好的线性规划算法所得的解相当。

实际上,对于采用Tabu搜索的EHM系统,当系统没有陷入一个角点时,式(4-67)的目标函数 $F_t(\hat{\mathbf{X}}, t)$ 中起主要作用的是 $F(\hat{\mathbf{X}})$ 这一项, $E(\hat{\mathbf{X}})$ 的梯度也由此项控制。一旦系统陷入了一个角点, $\mathcal{F}_t(\mathbf{X}, t)$ 就起主要作用,直至系统从陷阱中逃出再趋向另一角点。这样,在Tabu搜索中能够访问若干个 $F(\hat{\mathbf{X}})$ 取值较低的角点,从中可以找到符合约束条件(2)

且满足  $F(\hat{X})$  取较小值的解。而且系统所搜索的一些角点可能构成一个极限环，即这些角点可能被循环访问。

### 3. 目标函数的选择与约束面 $\varphi$ 的性质

各种 0-1 规划问题中有一些的目标函数只能有一种选择，例如背包问题的目标函数只具线性项而其约束面  $\varphi$  为非整的。再如图分割问题 graph partitioning problem, GPP)，其目标函数含二次项而  $\varphi$  为整约束面<sup>[41,42,43]</sup>。但是对于 TSP 而言，其目标函数可能有两种选择。一种选择是 Hopfield 所提出的方案，即目标函数含二次项，这时相应的  $\varphi$  为整约束面。另一种选择为目标函数只具线性项而  $\varphi$  为非整约束面<sup>[5,44]</sup>。可以看到 前一种方案的缺点是有多个局部极小，而其优点是不存在陷入不符合约束条件（2）的困境。后一方案的优缺点正好与前一方案相反。这样，在采用前一方案时必须采用 HC 和 / 或 SC 等算法 从而能逸出低质局部极小解。即便如此，当所处理的 TSP 不具欧氏性质时，解的质量仍不够满意。而在采用后一方案时不必用 HC 或 SC 算法 但必须采用 Tabu 搜索，以便从非整角点中逸出，这也使算法复杂度大为增加。

## 4.4 采用罚函数的 EHM

这一算法的思路是用等式和不等式约束（见式 4-2）和式 4-3）构成一个惩罚函数，当各项约束得到满足时，此罚函数达到其最小值。然后令此罚函数乘以某个适当的正系数后与待优化的目标函数（见式 4-1）相加 以形成 HM 的能量函数。当 HM 的动力学使得系统收敛到此能量函数的极小值点时，相应的解在满足各项约束的同时使目标函数达到极小。实际上，前文介绍的 Hopfield 等用 HM 求解 TSP 时所用的就是罚函数法。但是，其中各加权系数靠凑试来决定，而且解的可用性（指满足各项约束）和质量（指目标函数足够小）都没有保证。此外，它是一个解决特定问题的算法，特别是没有含不等式约束的项目，而不是一种解决一般 0-1 线性规划问题的算法。这一节介绍的 EHM 即在以上几方面对原 HM 予以改进。

### 4.4.1 由等式和不等式约束构成的罚函数，相应的能量函数和运行方程

根据 4.1 节给出的 0-1 规划问题，对于待求向量  $\mathbf{X} = [x_1, x_2, \dots, x_N]^T$  在满足式（4-2）~ 式（4-4）3 项约束条件下使目标函数  $F(\mathbf{X})$ （见式 4-1）达到最小。用式（4-2）的  $M$  项等式约束可形成一个罚函数  $G(\mathbf{X})$ ：

$$\left. \begin{aligned} G(\mathbf{X}) &= \frac{1}{2} \sum_{m=1}^M (\mathbf{a}_m^T \mathbf{X} - b_m)^2 - \frac{1}{2} \sum_{m=1}^M b_m^2 = \frac{1}{2} \mathbf{X}^T \mathbf{A} \mathbf{X} + \mathbf{b}^T \mathbf{X} \\ \mathbf{A} &= \sum_{m=1}^M \mathbf{a}_m \mathbf{a}_m^T, \quad \mathbf{b} = - \sum_{m=1}^M b_m \mathbf{a}_m \end{aligned} \right\} \quad (4-82)$$

可以看到，只有  $\mathbf{X}$  使  $m$  个等式约束条件  $\mathbf{a}_m^T \mathbf{X} = b_m, m = 1 \sim M$  都满足时  $G(\mathbf{X})$  达到其极小值。

为了用式（4-3）的  $K$  项不等式约束形成罚函数，则必须引入  $K$  个松弛变量  $y_k, k =$

1 ~ K。设  $\mathbf{Y} = [y_1, y_2, \dots, y_K]^T$  则罚函数  $H(\mathbf{X}, \mathbf{Y})$  为

$$H(\mathbf{X}, \mathbf{Y}) = \frac{1}{2} \sum_{k=1}^K (d_k y_k - \mathbf{r}_k^T \mathbf{X})^2 \quad (4-83)$$

其中各  $y_k$  的取值范围分成两种情况。

情况 1(上部受限)：

$$y_k \leq 1 \text{ 对应于 } \mathbf{r}_k^T \mathbf{X} \leq d_k \quad (4-84a)$$

情况 2(下部受限)：

$$y_k \geq 1 \text{ 对应于 } \mathbf{r}_k^T \mathbf{X} \geq d_k \quad (4-84b)$$

这样，如果某  $\mathbf{X}, \mathbf{Y}$  使得  $H(\mathbf{X}, \mathbf{Y})$  达到其极小值(此值为 0)且  $\mathbf{Y}$  各分量符合式(4-84)那么  $K$  项不等式约束都得到满足。作为第一步，这一节只讨论各  $d_k > 0$  的情况。更一般的  $d_k \geq 0$  的情况在 4.4.4 小节中讨论。式(4-83)可以改写成如下的展开形式：

$$H(\mathbf{X}, \mathbf{Y}) = \frac{1}{2} \{ \mathbf{X}^T \mathbf{R} \mathbf{X} + \mathbf{X}^T \mathbf{P} \mathbf{Y} + \mathbf{Y}^T \mathbf{P}^T \mathbf{X} + \mathbf{Y}^T \mathbf{D} \mathbf{Y} \} \quad (4-85)$$

其中  $\mathbf{R}, \mathbf{P}, \mathbf{D}$  分别是  $N \times N$  维、 $N \times K$  维、 $K \times K$  维矩阵，分别表示如下

$$\mathbf{R} = \sum_{k=1}^K \mathbf{r}_k \mathbf{r}_k^T, \mathbf{P} = -[d_1 \mathbf{r}_1, d_2 \mathbf{r}_2, \dots, d_K \mathbf{r}_K], \mathbf{D} = \text{diag}\{d_1^2, d_2^2, \dots, d_K^2\}$$

如果将  $\mathbf{X}$  和  $\mathbf{Y}$  并成一个  $N + K$  维列向量  $[\mathbf{X}^T, \mathbf{Y}^T]^T$  则式(4-85)更可以写成

$$H(\mathbf{X}, \mathbf{Y}) = \frac{1}{2} [\mathbf{X}^T, \mathbf{Y}^T] \begin{bmatrix} \mathbf{R} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \end{bmatrix} \quad (4-86)$$

现在能量函数含  $\mathbf{X}, \mathbf{Y}$  两个变向量且由目标函数  $F(\mathbf{X})$  和上述两个罚函数构成。这样，此能量函数应表示为  $E(\mathbf{X}, \mathbf{Y})$  其计算式为

$$\begin{aligned} E(\mathbf{X}, \mathbf{Y}) &= \alpha F(\mathbf{X}) + \beta G(\mathbf{X}) + \gamma H(\mathbf{X}, \mathbf{Y}) \\ &= \tilde{E}(\mathbf{X}) + \gamma H(\mathbf{X}, \mathbf{Y}) \quad \alpha, \beta, \gamma > 0 \end{aligned} \quad (4-87)$$

其中  $\tilde{E}(\mathbf{X})$  是由  $\alpha F(\mathbf{X})$  和  $\beta G(\mathbf{X})$  组成的，引用式(4-1)和式(4-82)它可以写成

$$\begin{aligned} \tilde{E}(\mathbf{X}) &= \frac{1}{2} \mathbf{X}^T (\alpha \mathbf{Q} + \beta \mathbf{A}) \mathbf{X} + (\alpha \mathbf{g} + \beta \mathbf{b})^T \mathbf{X} \\ &= \frac{1}{2} \mathbf{X}^T \tilde{\mathbf{W}} \mathbf{X} + \tilde{\mathbf{Z}}^T \mathbf{X} \end{aligned} \quad (4-88)$$

其中

$$\mathbf{W} = \alpha \mathbf{Q} + \beta \mathbf{A}, \mathbf{Z} = \alpha \mathbf{g} + \beta \mathbf{b}$$

可以看到，若无不等式约束，则式(4-88)给出的能量函数  $\tilde{E}(\mathbf{X})$  与式(4-17)给出的标准 HM 能量函数  $E(\mathbf{X})$  在形式上并无区别。若含不等式约束，则应增  $\gamma H(\mathbf{X}, \mathbf{Y})$  项。下面给出按此能量函数形成的 EHM 动力学。至于如何选择  $\alpha, \beta, \gamma$ ，使得系统所收敛到的稳定平衡角点必然满足各项约束条件，留待下一小节讨论。

此 EHM 含  $N + K$  个神经元 其中  $N$  个神经元的输出和输入用  $x_i$  和  $u_i$  表示  $i = 1 \sim N$  另外  $K$  个神经元的输出和输入用  $y_k$  和  $v_k$  表示  $k = 1 \sim K$  它们都是时间  $t$  的函数 为简洁起见，变量  $t$  都未标明。这些神经元的输入-输出关系由下式计算：

$$x_i = \frac{1}{2} \left[ 1 + \tanh \left( \frac{u_i}{T} \right) \right], \quad i = 1 \sim N, T > 0 \quad (4-89)$$

$$y_k = \begin{cases} 1 - \exp(-v_k/\rho), & \text{情况 1} \\ 1 + \exp(-v_k/\rho), & \text{情况 2} \end{cases}, \quad \rho > 0, k = 1 \sim K \quad (4-90)$$

可以看到,  $0 \leq x_i \leq 1$ ,  $-\infty < u_i < \infty$ ;  $y_k \leq 1$ (情况 1),  $y_k \geq 1$ (情况 2),  $-\infty < v_k < \infty$ 。为简化起见, 可以令  $T = \rho = 1$ , 下文均按此取值。

设  $U = [u_1, u_2, \dots, u_N]^T, V = [v_1, v_2, \dots, v_K]^T$  则它们随时间  $t$  的变化率可用下式计算:

$$\begin{bmatrix} \frac{dU}{dt} \\ \frac{dV}{dt} \end{bmatrix} = - \begin{bmatrix} \frac{\partial E(X, Y)}{\partial X} \\ \frac{\partial E(X, Y)}{\partial Y} \end{bmatrix} \quad (4-91)$$

将式 (4-86) 和式 (4-88) 代入式 (4-87) 得到

$$E(X, Y) = \frac{1}{2} [X^T, Y^T] \begin{bmatrix} W & \gamma P \\ \gamma P^T & \gamma D \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} + Z^T X \quad (4-92)$$

其中  $W = \tilde{W} + \gamma R, Z = \tilde{Z}$ 。将式 (4-92) 代入式 (4-91) 即得

$$\begin{bmatrix} \frac{dU}{dt} \\ \frac{dV}{dt} \end{bmatrix} = - \begin{bmatrix} W & \gamma P \\ \gamma P^T & \gamma D \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} - \begin{bmatrix} Z \\ 0 \end{bmatrix} \quad (4-93)$$

注意到

$$\begin{bmatrix} \frac{dX}{dt} \\ \frac{dY}{dt} \end{bmatrix} = \begin{bmatrix} \frac{\partial X}{\partial U} & \frac{\partial X}{\partial V} \\ \frac{\partial Y}{\partial U} & \frac{\partial Y}{\partial V} \end{bmatrix} \begin{bmatrix} \frac{dU}{dt} \\ \frac{dV}{dt} \end{bmatrix}$$

引用式 (4-89) 和式 (4-90) 可得

$$\left. \begin{aligned} \frac{\partial X}{\partial U} &= \begin{bmatrix} \frac{dx_1}{du_1} & 0 \\ & \frac{dx_2}{du_2} \\ & \ddots \\ 0 & \frac{dx_N}{du_N} \end{bmatrix} = \begin{bmatrix} 2x_1(1-x_1) & & \\ & 2x_2(1-x_2) & 0 \\ & \ddots & \\ 0 & & 2x_N(1-x_N) \end{bmatrix}, \frac{\partial X}{\partial V} = 0 \\ \frac{\partial Y}{\partial V} &= \begin{bmatrix} \frac{dy_1}{dv_1} & 0 \\ & \frac{dy_2}{dv_2} \\ & \ddots \\ 0 & \frac{dy_K}{dv_K} \end{bmatrix} = \begin{bmatrix} \varphi(y_1) & & \\ & \varphi(y_2) & 0 \\ & \ddots & \\ 0 & & \varphi(y_K) \end{bmatrix}, \frac{\partial Y}{\partial U} = 0 \end{aligned} \right\} \quad (4-94)$$

其中

$$\varphi(y_k) = \begin{cases} (1 - y_k), & \text{情况 } 1(y_k \leq 1) \\ (y_k - 1), & \text{情况 } 2(y_k \geq 1) \end{cases} \quad (4-95)$$

这样即可得罚函数 EHM 的运行方程如下：

$$\begin{bmatrix} \frac{d\mathbf{X}}{dt} \\ \frac{d\mathbf{Y}}{dt} \end{bmatrix} = - \begin{bmatrix} 2x_1(1-x_1) & & 0 \\ & \ddots & \\ & 2x_N(1-x_N) & \\ & \varphi(y_1) & \\ & \ddots & \\ 0 & & \varphi(y_K) \end{bmatrix} \left\{ \begin{bmatrix} \mathbf{W} & \gamma\mathbf{P} \\ \gamma\mathbf{P}^T & \gamma\mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \end{bmatrix} + \begin{bmatrix} \mathbf{Z} \\ \mathbf{0} \end{bmatrix} \right\} \quad (4-96)$$

如果  $\mathbf{X}, \mathbf{Y}$  是系统的解(平衡点)则应满足

$$\left. \frac{d\mathbf{X}}{dt} \right|_{(\mathbf{X}, \mathbf{Y})=(\hat{\mathbf{X}}, \hat{\mathbf{Y}})} = \mathbf{0}, \left. \frac{d\mathbf{Y}}{dt} \right|_{(\mathbf{X}, \mathbf{Y})=(\hat{\mathbf{X}}, \hat{\mathbf{Y}})} = \mathbf{0} \quad (4-97)$$

设  $\hat{\mathbf{X}} = [\hat{x}_1, \hat{x}_2, \dots, \hat{x}_N]^T, \hat{\mathbf{Y}} = [\hat{y}_1, \hat{y}_2, \dots, \hat{y}_K]^T$  则满足下列 3 个项目中任意一项者为系统的平衡点<sup>[10]</sup>

$$(1) \quad \hat{x}_i \in \{0, 1\}, i = 1 \sim N; \hat{y}_k = 1, k = 1 \sim K \quad (4-98a)$$

$$(2) \quad \mathbf{W}\mathbf{X} + \gamma\mathbf{P}\mathbf{Y} + \mathbf{Z} = \mathbf{0} \text{ 且 } \gamma\mathbf{P}^T\mathbf{X} + \gamma\mathbf{D}\mathbf{Y} = \mathbf{0} \quad (4-98b)$$

$$(3) \quad (1), (2) \text{ 两项皆满足。} \quad (4-98c)$$

事实上，系统还可能存在很多其他平衡点，这就是  $\mathbf{W}\mathbf{X} + \gamma\mathbf{P}\mathbf{Y} + \mathbf{Z}$  不等于  $\mathbf{0}$  而其中含若干 0 分量 对其非 0 分量则满足  $y_k = 1$ 。这样 就可以形成其他很多解。但是 我们最感兴趣的是解必须满足  $\hat{x}_i \in \{0, 1\}, i = 1 \sim N$  因为只有如此才满足 0-1 规划的策 3) 条约束。这种解称为角点，下一节研究如何设置  $\alpha, \beta, \gamma$  这些系数，使得满足 4.1 节所述 1)、(2) 两项约束（即等式和不等式约束）的角点是稳定平衡解；反之，则是不稳定平衡解。

#### 4.4.2 求系数 $\gamma, \alpha, \beta$ 的方法

基于式 (4-96) 的 EHM 运行方程，系统从任一初值出发将收敛到  $E(\mathbf{X}, \mathbf{Y})$  的一个局部极小点 也就是式 (4-96) 的一个稳定平衡解。为了使所得的解符合 4.1 节提出的 3 项关于 0-1 规划条件 这些解必须是  $\mathbf{X}$  空间中  $N$  维单位超立方体的角点且满足所有等式及不等式约束。因此 必须有一种系统地求  $\alpha, \beta, \gamma$  各系数的方法，这些系数使得满足所有等式及不等式约束的角点必然是系统的稳定平衡点，而任何不满足等式或不等式约束角点都是非稳定平衡点。这样，系统只要收敛到一个角点则必然是可用解（当然，还可能存在非角点的稳定平衡点，这些解不在关心之列）。

由于这 3 个待求系数之间的比例关系是决定因素，所以可以设  $\alpha = 1$  而令  $\beta, \gamma$  为待求者。下面首先讨论只有等式约束时如何求  $\beta$ 。其次将证明，如果一个角点满足不等式约束，那么该角点是否是稳定平衡点只取决于  $\tilde{E}(\mathbf{X})$  (见式 (4-88))，因此可以用只有等式约束

时的求  $\beta$  方法, 使得既满足不等式约束又满足等式约束的角点成为稳定平衡点。第三, 将给出求  $\gamma$  的方法, 使得不满足不等式约束的角点是非稳定平衡解。这样前面提的要求即可实现。

(1) 在只有等式约束条件下求  $\beta$ , 使得满足等式约束的角点一定是稳定平衡解。

在只有等式约束时, 由式 (4-87) 和式 4-82) 可得

$$E(\mathbf{X}, \mathbf{Y}) = \tilde{E}(\mathbf{X}) = F(\mathbf{X}) + \beta G(\mathbf{X}), G(\mathbf{X}) = \frac{1}{2} \sum_{m=1}^M (\mathbf{a}_m^T \mathbf{X} - b_m)^2 - \frac{1}{2} \sum_{m=1}^M b_m^2 \quad (4-99)$$

可以看到, 若有某个角点  $\tilde{\mathbf{X}}$  使得  $M$  项等式约束皆满足 即  $\mathbf{a}_m^T \tilde{\mathbf{X}} - b_m = 0, m = 1 \sim M$  这时  $G(\mathbf{X})$  在  $\tilde{\mathbf{X}}$  达到其极小值。若  $\tilde{\mathbf{X}}^{(i)}$  的第  $i$  分量与  $\tilde{\mathbf{X}}$  相反而其他分量相同, 则易于求得

$$\Delta G^{(i)}(\tilde{\mathbf{X}}) = G(\tilde{\mathbf{X}}^{(i)}) - G(\tilde{\mathbf{X}}) = \frac{1}{2} \sum_{m=1}^M a_{mi}^2 \quad (4-100)$$

在 4.2.4 小节中业已证明, 如果满足下列条件:

$$\tilde{E}(\tilde{\mathbf{X}}^{(i)}) - \tilde{E}(\tilde{\mathbf{X}}) > 0, \quad i = 1 \sim N \quad (4-101)$$

那么  $\tilde{\mathbf{X}}$  是以  $\tilde{E}(\mathbf{X})$  作为能量函数的 HM 系统的一个稳定平衡解 (见式 4-32))。

代入式 4-99) 此条件即可写成

$$[F(\tilde{\mathbf{X}}^{(i)}) - F(\tilde{\mathbf{X}})] + \beta [G(\tilde{\mathbf{X}}^{(i)}) - G(\tilde{\mathbf{X}})] > 0, \quad i = 1 \sim N \quad (4-102)$$

设所有角点的集合为  $\Gamma$  则必然有

$$F(\tilde{\mathbf{X}}^{(i)}) - F(\tilde{\mathbf{X}}) \geq \min_{\tilde{\mathbf{X}} \in \Gamma} [F(\tilde{\mathbf{X}})] - \max_{\tilde{\mathbf{X}} \in \Gamma} [F(\tilde{\mathbf{X}})]$$

而且

$$G(\tilde{\mathbf{X}}^{(i)}) - G(\tilde{\mathbf{X}}) \geq \min_{i=1 \sim N} [G(\tilde{\mathbf{X}}^{(i)}) - G(\tilde{\mathbf{X}})]$$

这样式 4-102) 可以重写成

$$\min_{\tilde{\mathbf{X}} \in \Gamma} [F(\mathbf{X})] - \max_{\tilde{\mathbf{X}} \in \Gamma} [F(\mathbf{X})] + \beta \cdot \min_{i=1 \sim N} [G(\tilde{\mathbf{X}}^{(i)}) - G(\tilde{\mathbf{X}})] > 0$$

由此即可求得参数  $\beta$  应满足

$$\beta > \frac{\max_{\tilde{\mathbf{X}} \in \Gamma} [F(\tilde{\mathbf{X}})] - \min_{\tilde{\mathbf{X}} \in \Gamma} [F(\tilde{\mathbf{X}})]}{\min_{i=1 \sim N} \left\{ \frac{1}{2} \sum_{m=1}^M a_{mi}^2 \right\}} \quad (4-103)$$

(2) 在具有不等式约束的情况下, 若角点  $\tilde{\mathbf{X}}$  满足全部不等式约束, 那么 EHM 系统在此  $\tilde{\mathbf{X}}$  处的特征值及稳定性只由  $\tilde{E}(\mathbf{X})$  来决定。

现在继续讨论基于运行方程式 (4-96) 的 EHM 系统, 设  $\hat{\mathbf{X}}, \hat{\mathbf{Y}}$  是其一组解 (平衡点), 见式 (4-97)。又设  $\tilde{\mathbf{X}}$  是一个角点 特别用  $\tilde{\mathbf{X}}$  表示 相应的  $\hat{\mathbf{Y}}$  也用  $\tilde{\mathbf{Y}}$  表示。如果在此点满足全部不等式约束, 由式 (4-83) 可知, 下列条件应成立:

$$H(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) = \frac{1}{2} \sum_{k=1}^K (d_k \tilde{y}_k - \mathbf{r}_k^T \tilde{\mathbf{X}})^2 = 0 \quad (4-104)$$

可以看到,  $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$  既然是  $H(\mathbf{X}, \mathbf{Y})$  的极小值点, 它必然满足

$$\left. \frac{\partial H(\mathbf{X}, \mathbf{Y})}{\partial \mathbf{X}} \right|_{\mathbf{X}=\tilde{\mathbf{X}}, \mathbf{Y}=\tilde{\mathbf{Y}}} = \mathbf{0} \quad (4-105)$$

将式 (4-85) 代入上式, 即可得

$$R\tilde{X} + P\tilde{Y} = 0 \quad (4-106)$$

下面讨论系统在  $(\mathbf{X}, \mathbf{Y})$  处的特征值。设  $\mathbf{X} = \tilde{\mathbf{X}} + \mathbf{X}'$ ,  $\mathbf{X}' = [x'_1, x'_2, \dots, x'_N]^T$ ,  $0 < |x'_i| \ll 1$  又设  $\mathbf{Y} = \tilde{\mathbf{Y}} + \mathbf{Y}'$ ,  $\mathbf{Y}' = [y'_1, y'_2, \dots, y'_K]^T$ ,  $0 < |y'_k| \ll 1$ 。 $\mathbf{X}'$  和  $\mathbf{Y}'$  是微小偏移向量。设式(4-92)公式中的  $\mathbf{W}$  为

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_1 \\ \mathbf{W}_2 \\ \vdots \\ \mathbf{W}_N \end{bmatrix}, \quad \mathbf{W}_i = [w_{i1}, w_{i2}, \dots, w_{iN}], \quad i = 1 \sim N \quad (4-107)$$

$\mathbf{P}$  由后文中式 4-113 给出,  $\mathbf{D}$  由式(4-85)给出 并且设  $\mathbf{Z} = [z_1, z_2, \dots, z_N]^T$ 。如果略去二阶微小量, 由式(4-96)可以导出  $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$  处  $\mathbf{X}'$  和  $\mathbf{Y}'$  的导数,

$$\left. \frac{d\mathbf{X}'}{dt} = \frac{d\mathbf{X}}{dt} \right|_{\mathbf{X}=\tilde{\mathbf{X}}+\mathbf{X}'} = \begin{bmatrix} 2(2\tilde{x}_1 - 1)(\mathbf{W}_1\tilde{\mathbf{X}} + \gamma\mathbf{P}_1\tilde{\mathbf{Y}} + z_1) & 0 \\ 2(2\tilde{x}_2 - 1)(\mathbf{W}_2\tilde{\mathbf{X}} + \gamma\mathbf{P}_2\tilde{\mathbf{Y}} + z_2) & \\ \vdots & \\ 0 & 2(2\tilde{x}_N - 1)(\mathbf{W}_N\tilde{\mathbf{X}} + \gamma\mathbf{P}_N\tilde{\mathbf{Y}} + z_N) \end{bmatrix} \mathbf{X}' \quad (4-108)$$

$$\left. \frac{d\mathbf{Y}'}{dt} = \frac{d\mathbf{Y}}{dt} \right|_{\mathbf{Y}=\tilde{\mathbf{Y}}+\mathbf{Y}'} = - \begin{bmatrix} \gamma\varphi(\tilde{y}_1)d_1^2 & 0 \\ \gamma\varphi(\tilde{y}_2)d_2^2 & \\ \vdots & \\ 0 & \gamma\varphi(\tilde{y}_K)d_K^2 \end{bmatrix} \mathbf{Y}' - \begin{bmatrix} \varphi(\tilde{y}_1) & 0 \\ \varphi(\tilde{y}_2) & \\ \vdots & \\ 0 & \varphi(\tilde{y}_K) \end{bmatrix} \mathbf{P}^T \mathbf{X}' \quad (4-109)$$

式(4-108)和式(4-109)可用特征值形式表示为

$$\frac{d\mathbf{X}'}{dt} = \begin{bmatrix} \lambda_{x1} & 0 \\ & \lambda_{x2} \\ & \vdots \\ 0 & \lambda_{xN} \end{bmatrix} \mathbf{X}', \quad \frac{d\mathbf{Y}'}{dt} = \begin{bmatrix} \lambda_{y1} & 0 \\ & \lambda_{y2} \\ & \vdots \\ 0 & \lambda_{yK} \end{bmatrix} \mathbf{Y}' + \Psi(\mathbf{X}') \quad (4-110)$$

其中  $\lambda_{xi}$  表示有不等式约束时角点  $\mathbf{X}$  处的特征值, 以与无不等式约束的特征值  $\lambda_i$  相区别。 $\lambda_{xi}$  和  $\lambda_{yk}$  分别为

$$\lambda_{xi} = 2(2\tilde{x}_i - 1)(\mathbf{W}_i\mathbf{X} + \gamma\mathbf{P}_i\mathbf{Y} + z_i), \quad i = 1 \sim N, \quad \lambda_{yk} = -\gamma\varphi(\tilde{y}_k) \cdot d_k^2, \quad k = 1 \sim K \quad (4-111)$$

$\Psi(\mathbf{X}')$  表示式(4-109)右侧第2项, 该项与  $\mathbf{Y}'$  无关。由式(4-95)可知,  $\varphi(\tilde{y}_k) \geq 0, \forall k$  且知  $\gamma > 0, d_k^2 > 0$  因此  $\lambda_{yk} \leq 0, \forall k$ 。可以证明, 各  $\lambda_{xi}$  与不存在不等式约束的情况下系统的特征值  $\lambda_i$  相同, 后者可由式(4-30)和式(4-88)求得 即有



$$\lambda_i = 2(2\tilde{x}_i - 1)(\mathbf{W}_i^T \mathbf{X} + \tilde{z}_i), \quad i = 1 \sim N \quad (4-112)$$

其中  $\tilde{\mathbf{W}}_i$  是式 (4-88) 中矩阵  $\tilde{\mathbf{W}}$  的第  $i$  行, 即

$$\tilde{\mathbf{W}} = \begin{bmatrix} \tilde{\mathbf{W}}_1 \\ \tilde{\mathbf{W}}_2 \\ \vdots \\ \tilde{\mathbf{W}}_N \end{bmatrix}$$

且  $\tilde{z}_i = z_i, \forall i = 1 \sim N$ 。设式 (4-86)、式 (4-92) 中的  $\mathbf{W}, \mathbf{R}$  和  $\mathbf{P}$  为

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_1 \\ \mathbf{W}_2 \\ \vdots \\ \mathbf{W}_N \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ \vdots \\ \mathbf{R}_N \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \vdots \\ \mathbf{P}_N \end{bmatrix} \quad (4-113)$$

则可得

$$\mathbf{W}_i = \mathbf{W}_i + \gamma \mathbf{R}_i, \quad i = 1 \sim N \quad (4-114)$$

这样, 式 (4-111) 可写成

$$\lambda_{xi} = 2(2\tilde{x}_i - 1)(\mathbf{W}_i^T \mathbf{X} + z_i) + 2(2\tilde{x}_i - 1)(\gamma \mathbf{R}_i^T \mathbf{X} + \gamma \mathbf{P}_i^T \mathbf{Y})$$

由式 (4-106) 和式 (4-113) 立即可得  $\mathbf{R}_i^T \tilde{\mathbf{X}} + \mathbf{P}_i^T \tilde{\mathbf{Y}} = 0, i = 1 \sim N$  即证明了上式右侧第 2 项为 0。由前文所述已知  $\mathbf{Z} = \tilde{\mathbf{Z}}$  即  $z_i = \tilde{z}_i$  因此证明了

$$\lambda_{xi} = \lambda_i, \quad i = 1 \sim N \quad (4-115)$$

下面将证明, 若  $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$  是式 (4-96) 的解 (平衡点)  $\tilde{\mathbf{X}}$  为角点并且这组解满足不等式约束, 且设  $\mathbf{X}^{(i)}$  为第  $i$  分量与  $\mathbf{X}$  的相反而其他分量相同的向量, 那么只要下式成立,  $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$  是稳定平衡解 (即平衡点的稳定性只由  $\tilde{E}(\mathbf{X})$  决定),

$$\tilde{E}(\tilde{\mathbf{X}}) < \tilde{E}(\mathbf{X}^{(i)}), \quad i = 1 \sim N \quad (4-116)$$

设  $(\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)})$  也是式 (4-96) 的解 (平衡点), 即下式成立:

$$\left. \frac{d\mathbf{X}}{dt} \right|_{(\mathbf{X}, \mathbf{Y}) = (\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)})} = \mathbf{0}, \quad \left. \frac{d\mathbf{Y}}{dt} \right|_{(\mathbf{X}, \mathbf{Y}) = (\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)})} = \mathbf{0}$$

由于  $\tilde{\mathbf{X}}^{(i)}$  也是角点, 上列第一式成立。而第二式是否成立取决于各不等式约束是否满足。若满足 则  $\mathbf{P}^T \tilde{\mathbf{X}}^{(i)} + \mathbf{D} \tilde{\mathbf{Y}}^{(i)} = \mathbf{0}$  (见式 (4-85) 和式 (4-83)) 且各  $\varphi(\tilde{y}_k^{(i)}) \geq 0 (\tilde{y}_k^{(i)} \geq 1 \text{ 或 } \tilde{y}_k^{(i)} \leq 1)$  若不满足 则  $\mathbf{P}^T \tilde{\mathbf{X}}^{(i)} + \mathbf{D} \tilde{\mathbf{Y}}^{(i)} \neq \mathbf{0}$ , 对于不满足不等式约束的那些项目  $k$  必然有  $\varphi(\tilde{y}_k^{(i)}) = 0 (\tilde{y}_k^{(i)} = 1)$ 。下面分两步证明式 (4-116) 的命题。

若  $(\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)}), i = 1 \sim N$ , 皆满足不等式约束, 那么相应的特征值  $\lambda_{xi}^{(i)}, i = 1 \sim N$ , 与没有不等式约束的特征值  $\lambda_i^{(i)}, i = 1 \sim N$  相同 (见式 (4-115)) 即

$$\lambda_{xi}^{(i)} = \lambda_i^{(i)}, \quad i = 1 \sim N \quad (4-117)$$

在 4.2.2 小节中已陈述, 对于没有不等式约束的情况, 即可根据各  $\lambda_i$  和  $\lambda_i^{(i)}$  的计算证明 若式 (4-116) 成立, 则  $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$  是稳定平衡解 (见式 (4-32))。

若各  $(\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)})$  中有若干个不满足不等式约束条件。则可以证明如下。

首先设置  $K$  个参数  $\alpha_k(\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)})$  来描述在  $(\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)})$  处  $K$  项不等式约束的违反度:

$$\alpha_k(\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)}) = d_k^2 \tilde{\mathbf{y}}_k^{(i)} - d_k \mathbf{r}_k^T \tilde{\mathbf{X}}^{(i)} = \begin{cases} 0, & \text{约束满足} \\ < 0, & \text{约束不满足 情况 1)} \\ > 0, & \text{约束不满足 情况 2)} \end{cases} \quad k = 1 \sim K, i = 1 \sim N \quad (4-118)$$

由这  $K$  个参数可形成下列不等式约束违反度向量：

$$\boldsymbol{\alpha}(\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)}) = [\alpha_1(\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)}), \dots, \alpha_K(\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)})]^T \quad i = 1 \sim N \quad (4-119)$$

由式 4-85) 和式 4-118) 可以导出

$$\boldsymbol{\alpha}(\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)}) = \mathbf{D}\tilde{\mathbf{Y}}^{(i)} + \mathbf{P}^T \tilde{\mathbf{X}}^{(i)} \quad i = 1 \sim N \quad (4-120)$$

其次，可以利用上列  $\boldsymbol{\alpha}(\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)})$  证明，在  $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$  处的特征值皆小于 0 即

$$\lambda_{xi} = \lambda_i < 0 \quad i = 1 \sim N \quad (4-121)$$

那么，在各个  $(\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)})$  处的特征值  $\lambda_{xi}^{(i)}$  皆大于 0。由式 4-111) 可得

$$\lambda_{xi}^{(i)} = 2(2\tilde{x}_i^{(i)} - 1)(\mathbf{W}_i \tilde{\mathbf{X}}^{(i)} + \gamma \mathbf{P}_i \tilde{\mathbf{Y}}^{(i)} + \mathbf{z}_i), \quad i = 1 \sim N \quad (4-122)$$

由于  $\tilde{x}_i^{(i)}$  和  $\tilde{x}_i$  相反，所以  $2\tilde{x}_i^{(i)} - 1 = -(2\tilde{x}_i - 1)$  再利用式 4-114) 则式 4-122) 可写成

$$\lambda_{xi}^{(i)} = -2(2\tilde{x}_i - 1)(\tilde{\mathbf{W}}_i \tilde{\mathbf{X}}^{(i)} + \mathbf{z}_i) - 2(2\tilde{x}_i - 1) \cdot \gamma(\mathbf{R}_i \tilde{\mathbf{X}}^{(i)} + \mathbf{P}_i \tilde{\mathbf{Y}}^{(i)}) \quad i = 1 \sim N \quad (4-123)$$

注意 前文已述 可用移项方法使  $\tilde{w}_{ii} = 0, i = 1 \sim N$  (见 4.2.3 节) 而  $\tilde{\mathbf{X}}^{(i)}$  与  $\tilde{\mathbf{X}}$  只有第  $i$  位不一致，因此  $\tilde{\mathbf{W}}_i \tilde{\mathbf{X}}^{(i)} = \tilde{\mathbf{W}}_i \tilde{\mathbf{X}}$ ，因而式 4-123) 右侧第 1 项即为 (见式 4-112)、式 (4-115))

$$-2(2\tilde{x}_i - 1)(\mathbf{W}_i \mathbf{X} + \mathbf{z}_i) = -\lambda_i = -\lambda_{xi}$$

式 (4-123) 右侧第 2 项可推导如下。由式 4-120) 可得  $\tilde{\mathbf{Y}}^{(i)} = \mathbf{D}^{-1} \boldsymbol{\alpha}(\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)}) - \mathbf{D}^{-1} \mathbf{P}^T \tilde{\mathbf{X}}^{(i)}$  则有

$$\mathbf{R}_i \tilde{\mathbf{X}}^{(i)} + \mathbf{P}_i \tilde{\mathbf{Y}}^{(i)} = \mathbf{R}_i \tilde{\mathbf{X}}^{(i)} + \mathbf{P}_i \mathbf{D}^{-1} \boldsymbol{\alpha}(\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)}) - \mathbf{P}_i \mathbf{D}^{-1} \mathbf{P}^T \tilde{\mathbf{X}}^{(i)}$$

由式 4-85) 可知

$$\mathbf{P}_i \mathbf{D}^{-1} \mathbf{P}^T = \left( \sum_{k=1}^K r_{ki} r_{k1} \sum_{k=1}^K r_{ki} r_{k2} \cdots \sum_{k=1}^K r_{ki} r_{kN} \right) = \mathbf{R}_i$$

因此式 (4-123) 右侧第 2 项等于  $-2(2\tilde{x}_i - 1) \gamma \mathbf{P}_i \mathbf{D}^{-1} \boldsymbol{\alpha}(\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)})$ 。这样就得到

$$\lambda_{xi}^{(i)} = -\lambda_i - 2(2\tilde{x}_i - 1) \gamma \mathbf{P}_i \mathbf{D}^{-1} \boldsymbol{\alpha}(\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)}) \quad (4-124)$$

再次引式 4-85) 和式 (4-119) 即得到

$$\lambda_{xi}^{(i)} = -\lambda_i + 2(2\tilde{x}_i - 1) \gamma \sum_{k=1}^K d_k^{-1} r_{ki} \alpha_k(\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)}), \quad i = 1 \sim N \quad (4-125)$$

由式 (4-121) 可知  $\lambda_i < 0, \forall i$  因此  $-\lambda_i > 0$ 。对于不等式约束的情况 1 当  $\tilde{\mathbf{X}}$  变成  $\tilde{\mathbf{X}}^{(i)}$  时设某项  $k$  不等式约束由满足变成不满足，即由  $\alpha_k(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) = 0$  变成  $\alpha_k(\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)}) < 0$  (见式 (4-118))。这说明 若  $\tilde{x}_i = 1$  且相应的  $\tilde{x}_i^{(i)} = 0$  时必定有  $r_{ki} < 0$  这时  $2\tilde{x}_i - 1) r_{ki} \alpha_k(\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)}) > 0$ 。对于任何  $k$  都可按此处理。已知  $d_k > 0$  这样式 (4-125) 右侧第 2 项中只要有一个不等式约束项不满足则其值必大于 0。对于不等式约束的情况 2 也可作同样分析并得到同样结论。这就可以得到如下结论：即使在各  $(\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)})$  处不等式约束不满足，只要  $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$  处的特征值  $\lambda_i < 0, i = 1 \sim N$  那么  $(\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{Y}}^{(i)})$  处的特征值  $\lambda_{xi}^{(i)} > 0, i = 1 \sim N$ 。因此类似于 4.2.2 小节中对于式 4-32) 的求证方法即可证明  $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$  是稳定平衡解。在 4.2.2 小节中业已证明  $\lambda_i < 0, i = 1 \sim N$  这个条件与  $\tilde{\mathbf{E}}(\tilde{\mathbf{X}}) < \tilde{\mathbf{E}}(\tilde{\mathbf{X}}^{(i)})$  这个条件等效。这

样就证实了式(4-116)的命题。既然满足不等式约束角点  $\tilde{\mathbf{X}}$  是否稳定平衡点只取决于  $\tilde{E}(\tilde{\mathbf{X}})$  (即只有等式约束时的能量函数)及相应的条件式(4-101),这时可通过控制参数  $\beta$  使得此点也满足等式约束时成为稳定平衡点(见式(4-103))。如果只具有不等式约束而不具有等式约束,即有  $\tilde{E}(\mathbf{X}) = F(\mathbf{X})$ ,那么当  $\tilde{\mathbf{X}}$  是满足不等式约束的角点时,只有当  $\tilde{\mathbf{X}}$  是  $F(\mathbf{X})$  的局部极小点时,它成为稳定平衡解。

(3) 当角点  $\tilde{\mathbf{X}}$  不满足不等式约束时,如何控制参数  $\gamma$  使其成为不稳定平衡点。

对于任何一个角点  $\tilde{\mathbf{X}}$  及相应  $\tilde{\mathbf{Y}}$  它的前  $N$  个关于  $\mathbf{X}$  的特征值  $\lambda_{xi}$  可以仿照式(4-125)的导出过程求得如下:

$$\lambda_{xi} = 2(2\tilde{x}_i - 1)(\tilde{\mathbf{W}}_i \tilde{\mathbf{X}} + z_i) - 2(2\tilde{x}_i - 1)\gamma \sum_{k=1}^K d_k^{-1} r_{ki} \alpha_k(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}), \quad i = 1 \sim N \quad (4-126)$$

其中  $\alpha_k(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) = d_k^2 \tilde{y}_k - d_k \mathbf{r}_k^T \tilde{\mathbf{X}}, k = 1 \sim K$  并且  $2(2\tilde{x}_i - 1)(\tilde{\mathbf{W}}_i \tilde{\mathbf{X}} + z_i) = \lambda_i, \lambda_i$  是在没有不等式约束的条件下系统在  $\tilde{\mathbf{X}}$  处的第  $i$  特征值。在本节策 1) 项中已给出,若  $\tilde{\mathbf{X}}$  满足等式约束便可以控制参数  $\beta$  使得所有  $\lambda_i < \alpha$  (见式(4-103))。这样,当  $\tilde{\mathbf{X}}$  不满足不等式约束时,为了使  $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$  成为非稳定平衡解,在各  $\lambda_{xi}$  中至少有 1 个大于 0。在实际中,  $r_{ki}$  这个参数非负,  $\gamma$  和  $d_k$  大于 0。若  $\alpha_k(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) < \alpha$  (情况 1, 见式(4-118))。如果  $\tilde{x}_i = 1$  则式(4-126)右侧第 2 项大于 0。只要  $\gamma$  选得足够大就可使此项之绝对值超过第 1 项,从而使  $\lambda_{xi} > 0$ 。是否有可能所有  $\tilde{x}_i \neq 1$  呢? 这只能有一个点:  $\tilde{\mathbf{X}} = \mathbf{0}$  即原点。对于  $\alpha_k(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) > 0$  (情况 2), 只要  $\tilde{x}_i = 0$  即可使此第 2 项大于 0。除了  $\tilde{\mathbf{X}}$  为全 1 向量外,总有分量为 0。这样即可确定除原点和全 1 角点以外,只要  $\gamma$  满足下式,即可使  $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$  成为非稳定平衡解:

$$\gamma > \frac{\max_{\tilde{\mathbf{X}} \in \varphi} |\tilde{\mathbf{W}}_i \tilde{\mathbf{X}} + z_i|}{\min_{\tilde{\mathbf{X}} \in \varphi} \left| \sum_{k=1}^K d_k^{-1} r_{ki} \alpha_k(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) \right|} \quad (4-127)$$

其中  $\varphi$  是所有满足等式约束的角点  $\mathbf{X}$  的集合。如果各  $d_k$  和各  $r_{ki}$  皆为整数,则  $\alpha_k(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$  对于不满足  $k$  项不等式约束的角点而言其最小值为  $d_k$ 。此外,在  $K$  项不等式约束中,最少可能只有 1 项约束被违反,即只有一项  $\alpha_k(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) \neq 0$ 。这时式(4-127)可用下式替代:

$$\gamma > \frac{\max_{\tilde{\mathbf{X}} \in \varphi} |\tilde{\mathbf{W}}_i \tilde{\mathbf{X}} + z_i|}{\min_{k=1 \sim K, i=1 \sim N, r_{ki} \neq 0} (r_{ki})} \quad (4-128)$$

总结起来,对于采用罚函数的 EHM 确定能量函数  $E(\mathbf{X}, \mathbf{Y})$  中 3 个关键参数  $\alpha, \beta, \gamma$  (见式(4-87))的方法是:令  $\alpha = 1$  用式(4-103)求  $\beta$  用式(4-128)求  $\gamma$ 。那么用这些参数构成的动力系统(其运行方程为式(4-96))所收敛的解(即式(4-96)的稳定平衡点)有什么特点呢?这可以分成以下三种情况来讨论。

(1) 既存在等式约束又存在不等式约束的情况。若  $\mathbf{X}$  是角点  $\tilde{\mathbf{X}}$  ( $\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}$  是式(4-96)的一个平衡点。当  $\tilde{\mathbf{X}}$  不满足不等式约束时,  $(\mathbf{X}, \mathbf{Y})$  一定不是一个稳定平衡点。这时  $K$  项不等式约束中有若干项不满足,其集合记为  $US$ 。那么可以知道,  $\tilde{y}_k = 1, \alpha_k(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) < 0$  (情况 1) 或  $> \alpha$  (情况 2) 当  $k \in US; \tilde{y}_k \leq 1$  (情况 1) 或  $\geq 1$  (情况 2),  $\alpha_k(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) = 0$  当  $k \notin US$ 。当  $\tilde{\mathbf{X}}$  满足不等式约束且满足不等式约束时,  $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$  是一个稳定平衡点,这时  $US$  是一个空集,即恒有  $\alpha_k(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) = 0, k = 1 \sim K$ 。若  $\tilde{\mathbf{X}}$  满足不等式约束但不满足等式约束时,或  $\mathbf{X}$  非角点

$\tilde{X}$  时,不能确定  $(\tilde{X}, \tilde{Y})$  或  $(X, Y)$  是不是稳定平衡点。这就是说可能存在一些稳定平衡点,它们是不满足等式约束(但满足不等式约束)的角点,或者是非角点。

(2) 只存在等式约束的情况,这相当于所有  $(\tilde{X}, \tilde{Y})$  都满足不等式约束。这时可引用(1)项中所有关于满足不等约束条件下的结论(包括  $X$  非角点  $X$  的情况)

(3) 只存在不等式约束的情况。若  $X$  是角点  $\tilde{X}$  且不满足不等式约束,这时  $(\tilde{X}, \tilde{Y})$  一定不是稳定平衡点且可引用(1)项有关分析。若  $X$  是角点  $\tilde{X}$  且满足不等式约束,这时  $(\tilde{X}, \tilde{Y})$  是否稳定平衡点只取决于  $X$  是否是目标函数  $F(X)$  的一个局部极小点。若是,(即式(4-30)或式(4-32)得到满足)则是稳定平衡点。若非,则是不稳定平衡点。若  $X$  非角点  $X$ ,则不能事先简单地确定其是否稳定平衡点。

从以上分析可以看到,采用这种  $\beta$  和  $\gamma$  的参数决定方法,不能保证系统所收敛的稳定平衡点一定是一个满足所有 3 项约束的可用解,对于上述策(3)项还不能保证使  $F(X)$  足够小的可用解一定是稳定平衡解。下面举一个例子来说明。

#### 4.4.3 从一个实例剖析采用罚函数的 EHM 所存在的问题

现在讨论 4.3.5.1 节给出的背包问题且只研究  $K = 1$  的情况,这时只存在一个不等式约束,即

$$\sum_{i=1}^N a_{1i} x_i \leq b_1$$

另一项约束是  $x_i \in \{0, 1\}, i = 1 \sim N$ 。在此实例中  $a_{1i}$  和  $b_1$  都是大于 0 的整数。现在需要求满足这两项约束的  $X = [x_1, x_2, \dots, x_N]^T$  使得价值函数  $C(X) = \sum C_i x_i$  达到最大,其

中  $C_i$  是大于 0 的整数。等效地是使目标函数  $F(X) = \sum g_i x_i$  达到最小,  $g_i = -C_i$ 。由于不存在等式约束且  $\alpha = 1$  则 EHM 的能量函数  $E(X, Y)$  为(见式(4-87))

$$E(X, Y) = F(X) + \gamma H(X, Y) = E(X) + \gamma H(X, Y)$$

其中  $\tilde{E}(X) = F(X) = \tilde{Z}^T X, \tilde{Z} = g = [g_1, g_2, \dots, g_N]^T$  (见式(4-88)) 即有  $\tilde{z}_i = g_i = -C_i, i = 1 \sim N$ 。式中的  $H(X, Y)$  可导出如下。首先,式(4-3)的不等式约束只有一项且为上部受限,即情况 1(见式(4-84a)) 相应地约束  $r_1^T X \leq d_1$  中,

$$r_1 = [r_{11}, r_{12}, \dots, r_{1N}]^T = [a_{11}, a_{12}, \dots, a_{1N}]^T \text{ 且 } d_1 = b_1.$$

由式(4-83)和式(4-86)可得

$$H(X, Y) = \frac{1}{2} (d_1 y_1 - r_1^T X)^2 = \frac{1}{2} [X^T, y_1] \begin{bmatrix} R & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} X \\ y_1 \end{bmatrix}$$

其中

$$R = \begin{bmatrix} R_1 \\ R_2 \\ \vdots \\ R_N \end{bmatrix}, \quad R_i = [r_{1i} r_{11}, r_{1i} r_{12}, \dots, r_{1i} r_{1N}], \quad P = [-d_1 r_{11}, -d_1 r_{12}, \dots, -d_1 r_{1N}]^T$$

系统的运行方程可由式(4-96)得到

$$\begin{bmatrix} \frac{d\mathbf{X}}{dt} \\ \frac{dy_1}{dt} \end{bmatrix} = - \begin{bmatrix} 2x_1(1-x_1) & & 0 \\ & \ddots & \\ 0 & & 2x_N(1-x_N) \\ & & & \varphi(y_1) \end{bmatrix} \left\{ \begin{bmatrix} \gamma\mathbf{R} & \gamma\mathbf{P} \\ \gamma\mathbf{P}^T & \gamma d_1^2 \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ y_1 \end{bmatrix} + \begin{bmatrix} \mathbf{Z} \\ 0 \end{bmatrix} \right\}$$

其中  $\varphi(y_1) = 1 - y_1$  (见式 (4-95) 情况 1),  $\gamma\mathbf{R} = \mathbf{W}$  (见式 (4-92)) 因  $\tilde{E}(\mathbf{X})$  只有线性项 所以  $\mathbf{W} = \mathbf{0}$ ,  $\mathbf{Z} = \mathbf{Z}$  (亦见式 (4-92))。易于证明 若各  $x_i$  和  $y_1$  的初值在  $(0, 1)$  区间中时, 按此式运行的结果是各  $x_i$  和  $y_1$  只能收敛在  $[0, 1]$  区间中。按此系统运行方程, 系统所收敛的稳定平衡解  $(\mathbf{X}, \hat{y}_1)$  必须满足下列两组条件。

### (1) 第 1 组 平衡点条件

这组条件由  $d\mathbf{X}/dt|_{\mathbf{X}=\hat{\mathbf{X}}, y_1=\hat{y}_1} = \mathbf{0}$  和  $dy_1/dt|_{\mathbf{X}=\hat{\mathbf{X}}, y_1=\hat{y}_1} = 0$  来确定, 即  $\hat{x}_i \in \{0, 1\}$  或  $[\gamma(\mathbf{R}_i\hat{\mathbf{X}} - r_{1i}d_1\hat{y}_1) + z_i] = 0, \forall i = 1 \sim N, \hat{y}_1 = 1$  或  $\gamma(\mathbf{P}^T\hat{\mathbf{X}} + d_1^2\hat{y}_1) = 0$ 。注意“或”, 不意味其左右两方条件互斥, 即允许它们同时成立。还注意, 这一组条件还意味着对于  $i = 1 \sim N$  可能某些编号  $i$  的  $\hat{x}_i \in \{0, 1\}$  而其他一些编号  $i$  的  $\hat{x}_i \notin \{0, 1\}$  对于后者必须满足

$$\gamma(\mathbf{R}_i\hat{\mathbf{X}} - r_{1i}d_1\hat{y}_1) + z_i = -\gamma r_{1i} \left( d_1\hat{y}_1 - \sum_{j=1}^N r_{1j}\hat{x}_j \right) + z_i = 0 \quad (4-129)$$

其中  $d_1\hat{y}_1 - \sum_{j=1}^N r_{1j}\hat{x}_j = d_1^{-1}\alpha_1(\hat{\mathbf{X}}, \hat{y}_1)$ , 是在  $(\mathbf{X}, \hat{y}_1)$  处的不等式约束不满足度 (见式 (4-118)) 由于讨论的是情况 1 必有  $\alpha_1(\hat{\mathbf{X}}, \hat{y}_1) \leq 0$ 。若  $z_i \neq 0$ , 为使上列式 (4-129) 的方括弧中项为 0 必然有  $\alpha_1(\mathbf{X}, \hat{y}_1) < 0$  即不等式约束不满足。这时对于  $\hat{y}_1$  项而言, 因为  $\mathbf{P}^T\hat{\mathbf{X}} + d_1^2\hat{y}_1 = d_1(d_1\hat{y}_1 - \sum_{j=1}^N r_{1j}\hat{x}_j) = \alpha_1(\hat{\mathbf{X}}, \hat{y}_1) \neq 0$  则必有  $\hat{y}_1 = 1$ 。这说明 对于本问题而言 若  $\hat{\mathbf{X}}$  不是角点  $\mathbf{X}$ , 则不等式约束一定不满足。

### (2) 第 2 组 稳定条件

若  $(\hat{\mathbf{X}}, \hat{y}_1)$  是一个平衡点, 它满足第 1 组条件。 $\hat{\mathbf{X}}$  的各个分量可分成两类。第一类各  $\hat{x}_i \in \{0, 1\}$  因此可以用  $\tilde{x}_i$  表示之。对于这些  $\hat{x}_i = \tilde{x}_i$  设  $x_i = \tilde{x}_i + x'_i$ ,  $x'_i$  是一微小变化量, 便能仿照式 (4-108) 导出  $x'_i$  随时间  $t$  的变化率为  $dx'_i/dt = \lambda_{\tilde{x}_i}x'_i$ 。其中  $\lambda_{\tilde{x}_i}$  可用式 (4-111) 计算 针对本实例可求得

$$\lambda_{\tilde{x}_i} = 2(2\tilde{x}_i - 1) \left[ -\gamma r_{1i} \left( d_1\hat{y}_1 - \sum_{j=1}^N r_{1j}\hat{x}_j \right) + z_i \right]$$

对这些  $\hat{x}_i = \tilde{x}_i$  而言 稳定条件是  $\lambda_{\tilde{x}_i} < 0$ 。第二类各  $\hat{x}_i \notin \{0, 1\}$  设  $x_i = \hat{x}_i + x'_i$ ,  $x'_i$  为微小变化量。由于在这一类  $\hat{x}_i$  上必须满足组 1) 中方括弧项的条件, 在忽略二阶微小量后即可求得

$$\frac{dx'_i}{dt} = -\hat{x}_i(1-\hat{x}_i)\gamma r_{1i}^2x'_i - \hat{x}_i(1-\hat{x}_i) \left[ \gamma r_{1i} \left( \sum_{\substack{j=1 \\ j \neq i}}^N r_{1j}x'_j - d_1y'_1 \right) \right]$$

对这类  $\hat{x}_i$  而言, 稳定性取决于上式右侧第 1 项  $x'_i$  前的系数  $-\hat{x}_i(1-\hat{x}_i)\gamma r_{1i}^2$  是否小于 0。易于看到 对本实例而言 由于  $0 < \hat{x}_i < 1, \gamma > 0$  此系数永远小于 0 因而其稳定性是永远得到保证的。如果所有  $\hat{x}_i = \tilde{x}_i, \mathbf{X}$  便是一个角点  $\mathbf{X}$ , 反之则否。

最后，为了保证所有不满足不等式约束的角点都是非稳定平衡点，由式 (4-128) 可算出参数  $\gamma$  应满足 (此例中  $\tilde{\mathbf{W}}_i = \mathbf{0}, \forall i = 1 \sim N$ )

$$\gamma > \max_{i=1 \sim N} |z_i| / \min_{i=1 \sim N} r_{1i}$$

下面讨论文献 [10] 给出的一个数字实例，此例中  $N = 6$  且目标函数和约束方程的参数为  $C_1 = 1, C_2 = 2, C_3 = 3, C_4 = 4, C_5 = 5, C_6 = 6, a_{11} = 3, a_{12} = 6, a_{13} = 1, a_{14} = 5, a_{15} = 4, a_{16} = 2, b_1 = 18$ 。由此可求得 EHM 参数为  $z_i = -C_i, r_{1i} = a_{1i}, i = 1 \sim 6, d_1 = b_1$ 。由此可算出  $\gamma > 6$  选  $\gamma = 7$ 。稍作检查就可以求得，此实例的最佳可用解是  $\hat{x}_1 = 0, \hat{x}_i = 1, i = 2 \sim 6$ 。但是，这个解并不是系统的稳定平衡点，因为可以求得在此点上  $\lambda_{x1} = -z_1 = 1 > 0, \lambda_{xi} = z_i < 0, i = 2 \sim 6$ 。可以发现除此角点外其他所有角点都不是稳定平衡点。特别是  $\hat{x}_i = 1, i = 1 \sim 6$  这个角点由于它违反不等式约束可以求得其特征值为  $\lambda_{xi} = 2[-7 \cdot r_{1i}(-3) + z_i] > 0, \forall i = 1 \sim 6$  因此它是非稳定的。这样只能进一步在非角点中找寻稳定平衡点。这种解难以尽求，下面给出其中的几个：

$$\begin{aligned} \hat{x}_1 &= 0.215873, \hat{x}_2 = 0.9, & \hat{x}_3 &= \hat{x}_4 = \hat{x}_5 = \hat{x}_6 = 1 \\ \hat{x}_1 &= 0.415873, \hat{x}_2 = 0.8, & \hat{x}_3 &= \hat{x}_4 = \hat{x}_5 = \hat{x}_6 = 1 \\ \hat{x}_1 &= 0.5, & \hat{x}_2 &= 0.7579365, \hat{x}_3 = \hat{x}_4 = \hat{x}_5 = \hat{x}_6 = 1 \\ \hat{x}_1 &= 0.615873, \hat{x}_2 = 0.7, & \hat{x}_3 &= \hat{x}_4 = \hat{x}_5 = \hat{x}_6 = 1 \\ \hat{x}_1 &= 0.7, & \hat{x}_2 &= 0.6579365, \hat{x}_3 = \hat{x}_4 = \hat{x}_5 = \hat{x}_6 = 1 \end{aligned}$$

例如，对于上列第一组解，可以证实上述 1) 的平衡点条件中的  $i = 1, 2$  两项由式 (4-129) 的方括弧项为 0 来保证而  $i = 3 \sim 6$  各项由  $\hat{x}_i \in \{0, 1\}$  来保证。而 2) 的稳定条件中的  $i = 1, 2$  两项由  $-\hat{x}_i(1 - \hat{x}_i)\gamma r_{1i}^2 < 0$  来保证，而  $i = 3 \sim 6$  各项则由各  $\lambda_{xi} < 0$  保证 ( $\lambda_{x3} = -2.6667, \lambda_{x4} = -1.3333, \lambda_{x5} = -3.6666, \lambda_{x6} = -5.3333$ )。其他解与此类同。事实上从  $\mathbf{X} = [0.15873, 1, 1, 1, 1, 1]^T$  这个点至点  $\mathbf{X} = [1, 0.5079365, 1, 1, 1, 1]^T$  可以画一条直线，在这直线上的任何一个点都是系统的稳定平衡点，它形成了一个收敛“峡谷”。上面列举的 5 个解都在此“峡谷”之中。至于每次运行所收敛的解是线上的哪一点或其他非角点则取决于初值设置。由于在此实例中，任何稳定平衡点（即系统每次运行所收敛的点必在其中）都非角点，因此任何解皆非可用解。如何才能将非可用解转换成可用解呢？一种方案是采取四舍五入的方案例如当  $\hat{x}_i < 0.5$  令其为 0  $\hat{x}_i > 0.5$  令其为 1。这样上述第 1 和第 2 个解就成为  $\hat{\mathbf{X}} = [0, 1, 1, 1, 1, 1]^T$  它与所需的最优解一致。但是第 4 和第 5 个解成为  $\mathbf{X} = [1, 1, 1, 1, 1, 1]^T$  这个解不符所需。而原第 3 个解则难以解释。在文献 [10] 中将此实例中的目标函数设置为  $F(\mathbf{X}) = \left( \sum_{i=1}^N C_i - \sum_{i=1}^N C_i x_i \right)^2$ ，从而形成一个二次目标函数。这时系统的所有角点仍然都不是稳定平衡点，而系统所收敛的解都不是角点且随  $\gamma$  的改变而变化。这样对于用罚函数的 EHM，存在着如何对不可用解作出解释的问题。从另一角度出发，就是能否有某种策略迫使解必须在角点中搜寻。这就是下一小节将介绍的采用竞争机制的罚函数 EHM。

#### 4.4.4 CAM-EHM

由上节可知，用罚函数 EHM 解决 0-1 规划问题时，对于只有等式约束或等式和不等

式约束皆具有的情况，问题较易解决，这时凡是可用解都一定是稳定平衡点。而对于只有不等式约束的情况，可用解不一定是稳定平衡点，这时必须从非角点的稳定平衡解中寻找替代者，这又面临着将非 0 非 1 的  $\bar{x}_i$  转换为 0 或 1 的难题。为了克服这一困难，文献[45]提出了一种基于竞争机制 (competitive activation mechanism, CAM) 的方案，即 CAM-EHM。这种方案的构思在于，当各项不等式约束未满足时，由其造成的对  $i = 1 \sim N$  各神经元的馈送应促成其趋向 1 或 0 而不是中间值。下面介绍它的原理，并且把上一小节  $d_k \neq 0, \forall k$  的限制扩展为允许  $d_k = 0$ 。

首先，将上一小节中对不等式约束所做的关于  $H(\mathbf{X}, \mathbf{Y})$ ,  $d_k$  和  $y_k$  的各项规定 (式 (4-83)、式 (4-84a) 和式 (4-84b)) 修正为

$$H(\mathbf{X}, \mathbf{Y}) = \frac{1}{2} \sum_{k=1}^K (d'_k y_k - \mathbf{r}_k^T \mathbf{X})^2 \quad (4-130)$$

其中  $d'_k = d_k$  当  $d_k \neq 0$ ;  $d'_k = 1$  当  $d_k = 0$ 。

情况 1(上部受限)：

$$\left. \begin{aligned} y_k &\leq 1, & \mathbf{r}_k^T \mathbf{X} &\leq d_k & \text{且} & d_k > 0 \\ y_k &\leq 0, & \mathbf{r}_k^T \mathbf{X} &\leq d_k & \text{且} & d_k = 0 \end{aligned} \right\} \quad (4-131a)$$

情况 2(下部受限)：

$$\left. \begin{aligned} y_k &\geq 1, & \mathbf{r}_k^T \mathbf{X} &\geq d_k & \text{且} & d_k > 0 \\ y_k &\geq 0, & \mathbf{r}_k^T \mathbf{X} &\geq d_k & \text{且} & d_k = 0 \end{aligned} \right\} \quad (4-131b)$$

这样，系统的能量函数  $E(\mathbf{X}, \mathbf{Y})$  (式 4-87)) 及其中各部分的计算公式与上一节全同。对于  $i = 1 \sim N$  诸神经元的输入  $u_i$  与输出  $x_i$  之间关系仍用式 4-89 计算。对于  $k = 1 \sim K$  诸神经元的输入  $v_k$  与输出  $y_k$  之间关系则改用下式计算：

$$y_k = \begin{cases} 1 - \exp(-v_k/\rho), & \text{情况 1 且 } d_k > 0 (y_k \leq 1) \\ -\exp(-v_k/\rho), & \text{情况 1 且 } d_k = 0 (y_k \leq 0) \\ 1 + \exp(v_k/\rho), & \text{情况 2 且 } d_k > 0 (y_k \geq 1) \\ \exp(-v_k/\rho), & \text{情况 2 且 } d_k = 0 (y_k \geq 0) \end{cases} \quad (4-132)$$

注意，在以下推导中皆取  $\rho = 1, T = 1$ 。

这样，系统的运行方程仍可用式 (4-96) 计算，只是其中的各  $\varphi(y_k)$  改用下式计算：

$$\varphi(y_k) = \begin{cases} 1 - y_k, & \text{情况 1 且 } d_k > 0 (y_k \leq 1) \\ -y_k, & \text{情况 1 且 } d_k = 0 (y_k \leq 0) \\ y_k - 1, & \text{情况 2 且 } d_k > 0 (y_k \geq 1) \\ y_k, & \text{情况 2 且 } d_k = 0 (y_k \geq 0) \end{cases} \quad (4-133)$$

且其中  $\mathbf{D} = \text{diag}\{(d'_1)^2, (d'_2)^2, \dots, (d'_K)^2\}$ 。参照式 (4-113)、式 (4-114)、式 (4-85) 和式 (4-97) 式 (4-96) 可以写成下列展开形式：

$$\frac{dx_i}{dt} = -2x_i(1-x_i)[\mathbf{W}_i \mathbf{X} + \gamma \mathbf{P}_i \mathbf{Y} + \mathbf{z}_i], \quad i = 1 \sim N \quad (4-134)$$

其中  $\mathbf{W}_i = \tilde{\mathbf{W}}_i + \gamma \mathbf{R}_i$ ,  $\mathbf{R}_i = \left( \sum_{k=1}^K r_{ki} r_{k1}, \sum_{k=1}^K r_{ki} r_{k2}, \dots, \sum_{k=1}^K r_{ki} r_{kN} \right)$   
 $\mathbf{P}_i = -[d'_1 r_{1i}, d'_2 r_{2i}, \dots, d'_K r_{Ki}]$ 。以及

$$\frac{dy_k}{dt} = -\varphi(y_k)[\gamma \mathbf{P}_k \mathbf{X} + \gamma (d'_k)^2 y_k], \quad k = 1 \sim K \quad (4-135)$$

其中  $\mathbf{P}_k = -[d'_k r_{k1}, d'_k r_{k2}, \dots, d'_k r_{kN}]$ 。式(4-134)和式(4-135)可以改写成下列形式：

$$\frac{dx_i}{dt} = 2(1-x_i) \left\{ \gamma x_i \left[ \sum_{k=1}^K r_{ki} \left( d'_k y_k - \sum_{j=1}^N r_{kj} x_j \right) \right] - x_i (\widetilde{\mathbf{W}}_i \mathbf{X} + z_i) \right\}, \quad i = 1 \sim N \quad (4-136)$$

$$\frac{dy_k}{dt} = -\gamma d'_k \varphi(y_k) \left( d'_k y_k - \sum_{j=1}^N r_{kj} x_j \right), \quad k = 1 \sim K \quad (4-137)$$

现在分析式(4-136)其右侧花括弧中的第1项可以写成

$$\gamma x_i \left[ \sum_{k=1}^K r_{ki} \left( d'_k y_k - \sum_{j=1}^N r_{kj} x_j \right) \right] = \gamma \sum_{k=1}^K x_i r_{ki} (d'_k)^{-1} \alpha_k(\mathbf{X}, \mathbf{Y}) \quad (4-138)$$

其中  $\alpha_k(\mathbf{X}, \mathbf{Y}) = d'_k (d'_k y_k - \mathbf{r}_k^T \mathbf{X})$  是在  $(\mathbf{X}, \mathbf{Y})$  处第  $k$  项约束的不满足度(见式(4-118)), 只需将其中  $d_k$  改成  $d'_k$ , 用  $\mathbf{X}, \mathbf{Y}$  替代  $\mathbf{X}^{(i)}, \mathbf{Y}^{(i)}$ 。这样, 当第  $k$  不等式约束不满足时,  $\alpha_k(\mathbf{X}, \mathbf{Y}) \neq 0$  它就会对  $i = 1 \sim N$  诸神经元馈送信号  $\zeta_{ki}$ 。

$$\zeta_{ki} = x_i r_{ki} (d'_k)^{-1} \alpha_k(\mathbf{X}, \mathbf{Y}) \quad (4-139)$$

可以将约束不满足度  $(d'_k)^{-1} \alpha_k(\mathbf{X}, \mathbf{Y})$  看成一项“资源”, 它在  $i = 1 \sim N$  诸神经元中进行分配以形成对其馈送信号  $\zeta_{ki}$ 。如有  $i, j$  两个神经元, 它们所接受的馈送信号  $\zeta_{ki}$  与  $\zeta_{kj}$  之比为

$$\left| \frac{\zeta_{ki}}{\zeta_{kj}} \right| \propto \frac{x_i}{x_j} \quad (4-140)$$

稍作分析就会发现这种“资源分配”方法对于各  $x_i$  收敛于1或0而非中间值是不利的。对于情况1(式(4-131a))当第  $k$  不等式约束不满足时,  $\alpha_k(\mathbf{X}, \mathbf{Y}) < 0$ 。若  $r_{ki} > 0, \forall i, k$  (下面均按此考虑)那么  $\zeta_{ki} < 0, \forall i = 1 \sim N$  此馈送信号迫使各  $x_i$  趋向于0从而使不等式约束  $k$  趋于满足。但是由式(4-140)可见若  $x_i > x_j$  则  $|\zeta_{ki}| > |\zeta_{kj}|$  这时幅度大的神经元较之幅度小的神经元所收到的负馈送信号幅度更大, 这种机制不能使原来有差异的  $x_i$  与  $x_j$  两个输出之间的差距拉开。最好的方法是用一种竞争机制, 即使那些本来已具有较小输出值的神经元在接收负馈送信号时具有更大优势, 这样就能迫使它们更快地趋向于0; 反之, 那些本来具有较大输出值的神经元在接收负馈送信号时具有弱势, 这样使它们较慢趋向于0。这样二者的差距可以拉开。在采用这种竞争机制时神经元  $i$  和  $j$  所接收馈送信号  $\zeta_{ki}$  与  $\zeta_{kj}$  之比应按下式计算。

$$\left| \frac{\zeta_{ki}}{\zeta_{kj}} \right| \propto \left( \frac{x_i}{x_j} \right)^\alpha, \quad 0 < \alpha < 1 \text{ (情况1)} \quad (4-141)$$

若  $x_j < x_i$  按此式计算时便得到  $|\zeta_{ki}/\zeta_{kj}| = (x_i/x_j)^\alpha < (x_i/x_j)$ 。这说明较之式(4-140)按此式求得的  $\zeta_{kj}$  相对于  $\zeta_{ki}$  要越大  $\alpha$  越接近0越大越多。这就能迫使幅度小的神经元更快地趋向于0。对于情况2(式(4-131b))当第  $k$  不等式约束不满足时  $\alpha_k(\mathbf{X}, \mathbf{Y}) > 0$  因此  $\zeta_{ki} > 0$ , 此馈送信号迫使各  $x_i$  趋向于1以使不等式约束  $k$  得到满足。但是按式(4-140)的资源分配方案, 不足以拉开两个已有差异的输出  $x_i$  和  $x_j$  之间的差距, 使得原来已接近于1的神经元接收更大的正馈送, 而原来接近于0的神经元接收较小的正馈送。为此,  $\zeta_{ki}$  与  $\zeta_{kj}$  之比应为

$$\left| \frac{\zeta_{ki}}{\zeta_{kj}} \right| \propto \left( \frac{x_i}{x_j} \right)^\alpha, \quad \alpha > 1 \text{ (情况2)} \quad (4-142)$$



若  $x_i > x_j$  按此式计算便得到  $|\zeta_{ki}/\zeta_{kj}| = (x_i/x_j)^a > (x_i/x_j)$ 。较之式 (4-140) 按此式求得的  $\zeta_{ki}$  相对于  $\zeta_{kj}$  要增大  $a$  越大, 增大越多。这就能迫使幅度大的神经元更快地趋向于 1。

为了实现上述的竞争机制, 可以按下述方法构成 CAM-EHM 其中必须满足  $d_k > 0$ ,  $\forall k = 1 \sim K$ 。此算法与普通罚函数 EHM 的区别是修正  $H(\mathbf{X}, \mathbf{Y})$ ,

$$H(\mathbf{X}, \mathbf{Y}) = \frac{1}{2} \sum_{k=1}^K \frac{\gamma_k}{\alpha_k} [d'_k y_k - \ln(\mathbf{r}_k^T \mathbf{X}^{(a_k)})]^2 \quad (4-143)$$

其中  $d'_k = \ln d_k$  当  $d_k \neq 1$ ;  $d'_k = 1$  当  $d_k = 1$ ;  $\mathbf{X}^{(a_k)} = [x_1^{a_k}, x_2^{a_k}, \dots, x_N^{a_k}]^T$ ;  $\gamma_k$  和  $\alpha_k$  皆大于 0 且可随  $k$  而改变。式中各  $y_k$  的变化范围以及  $y_k - v_k$  关系由下式描述:

$$y_k = \begin{cases} 1 - \exp(-v_k/\rho) & \text{情况 1 且 } d'_k \neq 0 (y_k \leq 1) \\ -\exp(-v_k/\rho) & \text{情况 1 且 } d'_k = 1 (y_k \leq 0) \\ 1 + \exp(v_k/\rho) & \text{情况 2 且 } d'_k \neq 0 (y_k \geq 1) \\ \exp(v_k/\rho) & \text{情况 2 且 } d'_k = 1 (y_k \geq 0) \end{cases} \quad (4-144)$$

在以下推导中取  $\rho = 1$ 。将此修正  $H(\mathbf{X}, \mathbf{Y})$  取代原 EHM 的  $H(\mathbf{X}, \mathbf{Y})$  见式 (4-83)) 并保持  $F(\mathbf{X})$  和  $G(\mathbf{X})$  不变, 就得到 CAM-EHM 的  $E(\mathbf{X}, \mathbf{Y})$  见式 (4-87)。依照 4.4.3 小节的推导, 即能得到此系统的运行方程,

$$\frac{dx_i}{dt} = 2(1 - x_i) \left\{ \sum_{k=1}^K \frac{\gamma_k r_{ki} x_i^{a_k}}{\mathbf{r}_k^T \mathbf{X}^{(a_k)}} [d'_k y_k - \ln(\mathbf{r}_k^T \mathbf{X}^{(a_k)})] - x_i (\tilde{\mathbf{W}}_i \mathbf{X} + z_i) \right\}, \quad i = 1 \sim N \quad (4-145)$$

$$\frac{dy_k}{dt} = -\frac{\gamma_k}{\alpha_k} d'_k \varphi(y_k) [d'_k y_k - \ln(\mathbf{r}_k^T \mathbf{X}^{(a_k)})], \quad k = 1 \sim K \quad (4-146)$$

注意 其中  $\mathbf{r}_k^T \mathbf{X}^{(a_k)} = \sum_{j=1}^N r_{kj} x_j^{a_k}$ ,  $\varphi(y_k)$  用式 (4-133) 计算。分析此运行方程可以发现系统有如下一些的特点。

(1) 对于第  $k$  项不等式约束不满足所造成的“资源”  $[d'_k y_k - \ln(\mathbf{r}_k^T \mathbf{X}^{(a_k)})]$  在  $i = 1 \sim N$  诸神经元之间按照竞争和归一化原则进行分配。即每个神经元由此资源所得的馈送信号  $\zeta_{ki}$  既符合式 (4-141) 或式 (4-142) 所描述的竞争机制, 又满足下列归一化条件:

$$\sum_{i=1}^N \zeta_{ki} = \frac{\sum_{i=1}^N r_{ki} x_i^{a_k}}{\mathbf{r}_k^T \mathbf{X}^{(a_k)}} \cdot \gamma_k [d'_k y_k - \ln(\mathbf{r}_k^T \mathbf{X}^{(a_k)})] = \gamma_k [d'_k y_k - \ln(\mathbf{r}_k^T \mathbf{X}^{(a_k)})] \quad (4-147)$$

(2) 对于情况 1 (上部受限) 由于按照式 (4-141)  $0 < \alpha_k < 1$  因此有  $x_i^{a_k} \geq x_i, i = 1 \sim N$ , 这表明

$$\mathbf{r}_k^T \mathbf{X}^{(a_k)} \geq \mathbf{r}_k^T \mathbf{X}$$

当不等式约束  $k$  得到满足时 实现的是

$$d_k \geq \mathbf{r}_k^T \mathbf{X}^{(a_k)}$$

而不是

$$d_k \geq \mathbf{r}_k^T \mathbf{X}$$

这说明, 用此方案求得的  $\mathbf{X}$  较之普通 EHM 被“低估”了。正是这种低估, 使得原算法中处于中间值的那些神经元被迫趋向于 0, 从而消除中间的含糊状态。

(3) 对于情况 2 下部受限),由式 (4-142)有  $\alpha_k > 1$  因此  $x_i^{\alpha_k} \leq x_i, i = 1 \sim N$  故

$$r_k^T X^{(\alpha_k)} \leq r_k^T X$$

当不等式约束  $k$  得到满足时,就有

$$d_k \leq r_k^T X^{(\alpha_k)} \leq r_k^T X$$

这说明,由此求得的  $X$ 较之原 EHM算法是被“高估”了,即各  $x_i$ 值偏大了,它使  $X$ 中有更多的 1 而较少中间含糊值。

(4) 参数确定。如果各  $r_{ki}$  和  $d_k$  都是非负整数,则 3 个关键参数  $\alpha, \beta, \gamma$  可确定如下。首先,令  $\alpha = 1$ 。其次  $\beta$  可用式 (4-103) 计算。文献 [45] 给出了如下一个更精确的估算公式。设  $X$  为角点,设  $\mathcal{Q}$  为各角点中各可用解的集合,则

$$\beta > \frac{\max_{\tilde{X} \in \mathcal{Q}} \max_{i=1 \sim N} [0, (F(\tilde{X}) - F(\tilde{X}^{(i)}))]}{\min_{i=1 \sim N} \left[ \frac{1}{2} \sum_{m=1}^M a_{mi}^2 \right]} \quad (4-148)$$

为此需要先求得  $\mathcal{Q}$  这很麻烦。最后,  $\gamma_k$  可用下式求得<sup>[45]</sup>:

$$\gamma_k > \frac{\max_{\tilde{X} \in \mathcal{Q}} \max_{i=1 \sim N} (F(\tilde{X}) - F(\tilde{X}^{(i)}))}{\frac{1}{2\alpha_k} [\ln(d_k * 1) - \ln d_k]^2} \quad (4-149)$$

其中对于情况 1 运算符“ $*$ ” = “+”,对于情况 2,“ $*$ ” = “-”。

下面介绍 CAM-EHM 的一个应用实例和模拟实验结果<sup>[45]</sup>。此实例为集合覆盖问题 (set covering problem),它可以抽象化为下列框架:设有  $N$  维向量  $X = [x_1, x_2, \dots, x_N]^T$ 。

目标函数为  $F(X) = \sum z_i x_i, z_i > 0, \forall i = 1 \sim N$ 。任务是在下列约束条件下求  $X$  使得  $F(X) \rightarrow \min$ 。

$$\text{约束条件 1: } \sum_{i=1}^N r_{ki} x_i \geq d_k, \quad k = 1 \sim K$$

其中  $r_{ki} \geq 0, d_k \equiv 1, \forall k = 1 \sim K$ 。

$$\text{约束条件 2: } x_i \in \{0, 1\}, \quad i = 1 \sim N$$

可以看到这是一个典型的下部受限 (情况 2) 不等式约束下的 0-1 规划问题且不存在等式约束。文献 [45] 针对此实例做了很多模拟实验。在一项模拟实验中  $N = 30, K = 60$ , 实验中取  $\alpha_k \equiv \alpha, \forall k = 1 \sim K$ 。由于是情况 2 且  $d_k \equiv 1$  故  $\gamma_k$  可取任意正数。在实验中用  $\alpha = 1, 2, 3$  三种值分别进行并且各  $x_i$  的初值取为 (0.45 ~ 0.55) 范围内的随机数。对系统所收敛的解按下列标准进行解释: 当  $x_i \leq 0.3$  即判定其为 0, 当  $x_i \geq 0.7$  即判定其为 1。当取  $\alpha = 1$  时, 进行了 3000 次迭代计算后, 仍有相当多的  $x_i$  处于 (0.3 ~ 0.7) 的模糊区中, 无法作出解释。但是较之用普通罚函数 EHM 的实验结果要改善很多。当  $\alpha = 2$  时, 只需进行 1000 次迭代计算, 则每一个  $x_i$  都得到了明确解释。当  $\alpha = 3$  时, 只需进行 100 次迭代即可得明确解。但是解的质量不如  $\alpha = 2$  的情况。此外, 还针对不同的  $N$  和  $K$  做了实验并与常用的启发式算法进行比较。结论是, 无论从解的质量还是所需的计算时间比较, CAM-EHM 皆优于启发式算法。需要指出的是, 即使采用了 CAM 策略, 当  $N$  和  $K$  较大时, 实验结果中也会出现若干个  $x_i$  处于模糊的中间态, 其数量不超过 3 ~ 5 个 (上限为 8 个)。

#### 4.4.5 MFT

MFT 是平均场理论 (mean field theory) 的缩写, 也称为 MFA (mean field annealing)<sup>[46,47]</sup>。以背包问题为例, 其基本构思如下。

对于 4.3.5.1 小节中描述的背包问题, 可以概括为在式 (4-63) 和式 4-64) 的约束条件下求  $\mathbf{X}$ , 使  $F(\mathbf{X}) = -\sum_{i=1}^N C_i x_i$  达到极小。为此, 可构成如下能量函数  $E(\mathbf{X})$ :

$$E(\mathbf{X}) = F(\mathbf{X}) + \gamma H(\mathbf{X}), \quad H(\mathbf{X}) = \sum_{k=1}^K \Phi\left(\sum_{i=1}^N a_{ki} x_i - b_k\right) \quad (4-150)$$

其中函数  $\Phi(\cdot)$  由下式计算:

$$\Phi(S) = S \cdot \theta(S), \quad \theta(S) = \begin{cases} 1, & S \geq 0 \\ 0, & S < 0 \end{cases} \quad (4-151)$$

可以看到,  $H(\mathbf{X})$  是一个罚函数, 当不等式约束都得到满足时  $H(\mathbf{X}) = 0$ ; 反之,  $H(\mathbf{X}) > 0$  且约束不满足度越高,  $H(\mathbf{X})$  越大。这样, 当  $\gamma$  取得充分大时, 在  $E(\mathbf{X})$  的各个极小值点上不等式约束应得到满足。这个罚函数与 4.4.4 小节所取罚函数的明显区别是此处未引入松弛变量  $y_k, k = 1 \sim K$ , 而且函数取为分段线性函数而不是二次函数。

下面给出一种迭代算法框架, 使得按节拍  $t = 0, 1, \dots$  对  $x_i(t)$  进行迭代运算时,  $x_i(t), i = 1 \sim N$  将逐渐收敛到  $E(\mathbf{X})$  的一个高质量局部极小点, 在此点上  $E(\mathbf{X})$  足够小。

如给定初值  $x_i(0)$  则对于任何  $t \geq 0$  由  $x_i(t)$  求  $x_i(t+1)$  的计算公式为

$$\begin{aligned} x_i(t+1) &= \frac{1}{2} \left[ 1 + \tanh\left(-\frac{\partial E(\mathbf{X})}{\partial x_i} \bigg|_{\mathbf{X}=\mathbf{X}(t)} \cdot \frac{1}{T_t}\right) \right] \\ &= \frac{1}{1 + \exp\left(\frac{1}{T_t} \cdot \frac{\partial E(\mathbf{X})}{\partial x_i} \bigg|_{\mathbf{X}=\mathbf{X}(t)}\right)}, \quad i = 1 \sim N \end{aligned} \quad (4-152)$$

其中

$$\begin{aligned} \frac{\partial E(\mathbf{X})}{\partial x_i} \bigg|_{\mathbf{X}=\mathbf{X}(t)} &= -C_i + \gamma \sum_{k=1}^K \left[ \Phi\left(\sum_{j=1}^N a_{kj} x_j - b_k\right) \right]_{\substack{x_j = x_j(t), j \neq i \\ x_i = 1}} - \\ &\quad \Phi\left(\sum_{j=1}^N a_{kj} x_j - b_k\right) \bigg|_{\substack{x_j = x_j(t), j \neq i \\ x_i = 0}}, \quad i = 1 \sim N \end{aligned} \quad (4-153)$$

实际上, 此式右侧只是其左侧的近似值。采用近似的原因是  $\Phi(\cdot)$  为非可微函数以及计算的便利。式 4-122) 中的  $T_t$  是一个随  $t$  而改变的退火温度。当  $t$  值较小时  $T_t$  取很大值, 这时各  $x_i(t)$  都在 1/2 附近取值。当  $t$  增大时  $T_t$  逐渐减小, 这时各  $x_i$  趋向于 0 或 1, 并且随着  $T_t \rightarrow 0$  各  $x_i$  只能取 0 或 1 值。如果温度  $T_t$  降得足够慢, 就可以使最终收敛到一个高质量局部极小点 (即使不是全局极小点)。

下面讨论运行此算法的一些具体问题。

##### (1) 降温准则

文献[47] 给出的准则是, 初始温度  $T_0 = 10^\circ\text{C}$ ; 对于  $t > 0, T_t$  由下式计算:

$$T_t = \zeta_{t-1} T_{t-1}, \quad \zeta_{t-1} = \begin{cases} 0.99, & 0.1 < \Sigma_{t-1} < \frac{N-1}{N} \\ 0.9, & \Sigma \text{ 为其他值} \end{cases} \quad (4-154)$$

其中

$$\Sigma_{t-1} = \frac{4}{N} \sum_{i=1}^N (x_i(t-1) - 0.5)^2 \quad (4-155)$$

$\Sigma$  描述系统的“饱和度”。当各  $x_i = 0.5$  时  $\Sigma = 0$  这时最不饱和。当  $x_i \in \{0, 1\}$  时  $\Sigma = 1$ , 这时系统最饱和。此外, 令参数  $\gamma$  亦随时间  $t$  而变化, 记为  $\gamma_t$ ,

$$\gamma_t = 1/T_t \quad (4-156)$$

这样, 可以保证当迭代过程后期  $\gamma_t$  足够大, 从而不等式约束能够得到满足。

## (2) 迭代终止准则

设置参数  $\epsilon$  如下:

$$\Delta_t = \frac{1}{N} \sum_{i=1}^N [x_i(t+1) - x_i(t)]^2 \quad (4-157)$$

若  $\Sigma_t > 0.999$  且  $\Delta_t < 0.00001$  则迭代计算结束。

文献[46]、[47]给出了对于不同规模和不同性质的背包问题采用 MFT 算法与采用其他算法的实验结果的比较。当  $N, K$  取值在 20 ~ 40 范围内时为小规模问题, 当取值在 100 以上时为大规模问题。当约束方程中各  $a_{ki}$  为  $[0, 1]$  中均匀分布的随机数时, 若  $b_k \approx \frac{N}{2}$

且  $N$  值较大时, 几乎所有  $x_i$  皆取 1; 若  $b_k \ll \frac{N}{4}$ , 则只有少数几个  $x_i$  取 1。对于这两种情况,

优化问题易于解决。若  $b_k \approx \frac{N}{4}$ , 则各  $x_i$  中应该约有一半取 1, 另一半取 0 值, 这是最难解决

的情况。在模拟实验中只考虑此最难情况。若目标函数中各  $C_i$  分布在一个很窄的范围内, 例如  $[0.45, 0.55]$  甚至  $C_i = C, \forall i = 1 \sim N$  这种情况称为“齐次”的。若  $C_i$  分布在一个宽范围内例如  $[0, 1]$  则称为“非齐次”的。由于非齐次情况有额外信息可资利用, 较齐次情况易于解决。除了 MFT 算法外, 现在已有各种启发式算法, 如 BB 和 GH (见文献[46])。另外, 还有一种基于单纯型法的线性规划算法, 记为 LP<sup>[48]</sup>。后者所求得各  $x_i$  有若干是 1 或 0 而尚余留一部分为中间范围值。对于这些剩余未决者尚可用 MFT 或 GH 算法将其值判为 0 或 1 从而构成 LM 或 LG 算法。

模拟实验结果表明, 当问题的规模越大以及问题的难度 (指是否齐次) 越大时, MFT 算法以及 LP 和 MFT 相结合而构成的 LM 算法的优势越大。例如, 对于齐次问题, 当  $N = K = 50$  时, 无论用 MFT、LM、LG 还是模拟退火 (SA) 算法<sup>[47]</sup>, 所得解的质量及所耗费的计算机时间都相差不大。前者以收敛解  $\mathbf{X}$  的价值函数  $C(\mathbf{X}) = \sum_{i=1}^N C_i x_i$  之值的大小来衡量, 后者以达到收敛所需的 CPU 时间 (以秒计) 来衡量。而当  $N = K = 400$  时, 模拟结果见表 4-4。

表 4-4 各算法的模拟结果

算法	MFT	LM	LG	SA
$C(\hat{X})$	190.0	193.1	186.1	187.1
CPU 时间 /s	79	1054	1026	57

此外, MFT 和相应的 LM 算法还可以用 VLSI 芯片来构成硬件实现, 其计算速度就更快了。而且模拟实验表明<sup>[47]</sup>, 当运算和权的精度取为 6bit 时, 计算结果几乎不受影响。

在以上诸节中主要讨论的是一个相当困难的优化问题, 即具有等式约束和不等式约束的 0-1 规划问题。然而在科学技术领域有相当多实际的问题是较易解决的只具有等式约束的 0-1 规划问题。这些问题用 EHM 来求解时可以得到很好的效果, 因此受到广泛重视。在通信和信号处理领域就可以举出很多应用实例, 由于篇幅的限制这里不详述其细节, 下面只列举其中若干课题及有关的参考文献。在通信领域中的应用可以举出: (1) 卫星广播排序<sup>[49]</sup>, (2) 通信系统控制<sup>[50]</sup>, (3) 计算机网络最短路径计算和路由管理<sup>[51]</sup> (4) ATM 和互联网中路由控制, 话务管理 (traffic management) 和组播路由 (multicasting) 控制<sup>[52~57]</sup>, (5) 电话网话务管理<sup>[58]</sup>, (6) CDMA 多用户检测接收机<sup>[59]</sup>。在其他领域的应用还可以举出: 自适应图像分割<sup>[60]</sup> 故障诊断<sup>[61]</sup> 以及实时交通系统的过程排序<sup>[62]</sup> 等。

最后, 还要指出, HM 还有很多其他改进的方法。最新提出的方法包括具有滞后效应 (hysteretic) 神经元函数的方案<sup>[63]</sup> 以及采用扩充拉格朗日乘子 (augmented Lagrange multiplier) 的方案<sup>[64]</sup>, 它们可以更有效地避免陷入低质局部极小的问题。

## 4.5 离散时间 HNN 与自联想记忆

离散时间 HNN (为简便计, 下文中只称之为 HNN) 由  $N$  个神经元构成, 每个神经元的输出用  $x_i$  表示, 所有输出构成一个  $N$  维列向量  $\mathbf{X} = [x_1, x_2, \dots, x_N]^T$ 。每个  $x_i$  的取值只能是  $-1$  或  $+1$  这可以表示为  $x_i \in \{-1, 1\}, \forall i = 1 \sim N$ 。HM 与 HNN 的区别是前者的各  $x_i$  在运行过程中允许取连续值, 只是在最后才收敛到离散的 0 或 1 值且运行的时间变量  $t$  取连续值, 而后者的各  $x_i$  无论在运行过程中还是最后收敛值都只能取为  $-1$  或  $+1$  且时间变量  $t$  取离散值。HNN 的运行方程是, 如果时刻  $t$  的各神经元输出为  $x_i(t)$ ,  $i = 1 \sim N$  那么  $t+1$  时刻的各输出由下列两种方式来决定。

### (1) 同步方式

每个时刻  $t$  所有神经元同时调整, 即

$$x_i(t+1) = \operatorname{sgn} \left[ \sum_{j=1}^N w_{ij} x_j(t) \right], \quad i = 1 \sim N \quad (4-158)$$

其中

$$\operatorname{sgn}[u] = \begin{cases} 1, & u \geq 0 \\ -1, & u < 0 \end{cases} \quad (4-159)$$

$w_{ij}$  为由神经元  $j$  至神经元  $i$  的连接权。各  $w_{ij}$  构成一个  $N \times N$  维权矩阵  $\mathbf{W}$  在 HNN 中规定  $\mathbf{W}$  是一个 0 主对角线对称阵, 即满足下列条件:

$$w_{ij} = w_{ji}, \quad \forall i, j = 1 \sim N, \quad w_{ii} = 0, i = 1 \sim N \quad (4-160)$$

## (2) 非同步方式

每个时刻  $t$  只有一个神经元得到调整, 即

$$x_i(t+1) = \begin{cases} \operatorname{sgn} \left[ \sum_{j=1}^N w_{ij} x_j(t) \right], & i = l(t) \\ x_i(t), & i \neq l(t) \end{cases} \quad (4-161)$$

其中  $l(t)$  是  $t$  时刻被选中的一个神经元标号, 一般按依次轮选方式, 例如  $l(1) = 1$ ,  $l(2) = 2, \dots, l(N) = N, l(N+1) = 1, \dots$ 。其他的方式包括随机选择或按照某种特定要求来选择。调整式中各  $w_{ij}$  构成的矩阵  $\mathbf{W}$  与同步方式的规定一致。

HNN 在时刻  $t$  的  $\mathbf{X}$  取值  $\mathbf{X}(t)$  称为其“状态”(state) 对任何状态  $\mathbf{X}$  可以定义一个相应的能量函数  $E(\mathbf{X})$ ,

$$E(\mathbf{X}) = -\frac{1}{2} \mathbf{X}^T \mathbf{W} \mathbf{X} \quad (4-162)$$

这是一个 Liapunov 函数。当网络的权矩阵  $\mathbf{W}$  给定以后, 若从某个初始状态  $\mathbf{X}(0)$  出发 按照节拍  $t = 0, 1, 2, \dots$  根据式 (4-158) 或式 (4-161) 运行网络, 可以证明  $\mathbf{X}(t)$  必然收敛到一个稳定状态, 达到此稳定状态后, 无论按同步还是非同步方式继续运行网络时, 其状态保持恒定不变<sup>[66]</sup>。HNN 的稳定状态称为“吸引子”(attractor) 或称为“定点”(fixed point)。根据式 (4-158) 和式 (4-161) 可知, 某一状态成为吸引子或定点的条件是

$$x_i = \operatorname{sgn} \left[ \sum_{j=1}^N w_{ij} x_j \right], \quad i = 1 \sim N \quad (4-163)$$

当某个时刻  $t_a$  网络状态  $\mathbf{X}(t)$  达到某个吸引子, 即符合式 (4-163) 条件时, 无论按同步或非同步方式继续运行, 都永远有

$$\mathbf{X}(t+1) = \mathbf{X}(t) \quad t \geq t_a$$

如果 HNN 有多个吸引子, 那么网络收敛到其中哪一个取决于初始状态  $\mathbf{X}(0)$  对于此问题将在后文中讨论。

只要具备一些弱约束条件 就可以证明 若某个  $\mathbf{X}$  是吸引子, 则它必是  $E(\mathbf{X})$  的一个局部极小点 反之亦对。证明见文献 [66]、文献 [67] 此处不再赘述。

HNN 的主要功能是实现联想记忆。4.5.1 小节将介绍实现联想记忆时的一些主要要求和基本定义以及用 Hopfield 所提出的原型网络实现时所达到的水平。4.5.2 小节介绍从改变神经元函数着手进行改进。4.5.3 小节介绍一种沿“最小覆盖”思路设计 HNN 的方案。4.5.4 小节介绍一种改变学习算法的改进方案。

### 4.5.1 联想记忆的容量、吸引域、记忆提取及其标准 HNN 实现

设有  $M$  个记忆项:  $\mathbf{X}^{(m)} = [x_1^{(m)}, x_2^{(m)}, \dots, x_N^{(m)}]^T, m = 1 \sim M$ 。其中各  $x_i^{(m)} \in \{-1, 1\}$ ,  $\forall i = 1 \sim N, m = 1 \sim M$ 。现在用一个 HNN 来存储这  $M$  个记忆项。按照 Hopfield 提出的原型方案<sup>[1]</sup> 此 HNN 的权矩阵  $\mathbf{W}$  由下式决定:

$$\mathbf{W} = \frac{1}{N} \sum_{m=1}^M [\mathbf{X}^{(m)} (\mathbf{X}^{(m)})^T - \mathbf{I}_N] \quad \text{或} \quad \mathbf{W} = \frac{1}{M} \sum_{m=1}^M [\mathbf{X}^{(m)} (\mathbf{X}^{(m)})^T - \mathbf{I}_N] \quad (4-164)$$

其中  $I_N$  是一个  $N \times N$  维单位阵。此计算式称为外积 (outer-product) 和  $W$  满足式 (4-160) 的条件且符合 Hebb 学习律<sup>[65]</sup> 当各记忆项中  $x$  和  $x_j^{(m)}$  具有相同符号的次数较多时,  $w_{ij}$  或  $w_{ji}$  取较大正值 反之 则取较负的负值。式 (4-164) 给出的就是此标准 HNN 的记忆存储或记忆学习算法。 $r = M/N$  称为此记忆器的装载率 (loading rate)。下面讨论记忆的提取 (recall)。如果有一个某记忆项  $X^{(m)}$  的受损或受污染的样本  $\hat{X}^{(m)}$  所谓“受损”或“受污”是指  $\hat{X}^{(m)}$  与原  $X^{(m)}$  之间有若干个分量不一致 (1 变成 -1 或反之) 现可以用  $\hat{X}^{(m)}$  从 HNN 中提取出  $X^{(m)}$  令  $t = 0$  时 HNN 的状态  $X(0) = \hat{X}^{(m)}$  如 HNN 按同步方式运行 则可由式 (4-158) 求得  $t = 1, 2, \dots$  诸时刻的  $X(t)$ 。若  $X(1) = X^{(m)}$  则称为记忆的一步 (直接) 提取; 若需经过多步运行后系统才能收敛到  $X^{(m)}$  即  $X(t) = X^{(m)}, t \geq 2$  这称为多步 (间接) 提取。易于看到, 成功实现提取任一记忆项的前提是每一个记忆项  $X^{(m)}$  必须是 HNN 的吸引子, 即满足式 (4-163) 的条件, 否则 HNN 不能收敛并稳定在所需的记忆项上。此条件可描述为

$$x_i^{(m)} = \operatorname{sgn} \left[ \sum_{j=1}^N w_{ij} x_j^{(m)} \right], \quad i = 1 \sim N, m = 1 \sim M \quad (4-165)$$

如 HNN 按非同步方式运行, 那么只有通过多步运行才有可能收敛到所需的解上。现在立即可以提出两个问题: 第一, 如何保证式 (4-165) 成立 第二  $\hat{X}^{(m)}$  与  $X^{(m)}$  的最大差异允许值是多少, 以保证一步收敛或多步收敛? 这两方面都取决于装载率  $r$  的大小且与  $M$  个存储项的性质有关。为使问题简化, 本节与 4.5.2 节中只研究各存储项相互独立且每个存储项中各分量相互独立的情况, 一般假设每个分量取 1 或 -1 的概率都是 1/2。在 4.5.3 小节和 4.5.4 小节中将包含各存储项具有相关性的情况。

第一个问题 在 Hopfield 的经典研究<sup>[1]</sup> 中给出的答案是当  $N$  充分大时 若  $r \leq 0.15$  则式 (4-165) 成立。在 Hopfield 工作的基础上, 大量模拟实验和理论分析都表明, 当时很小时, 式 (4-165) 总能成立, 当  $r$  增加并超过某个上限  $R$  时, 该式不再成立。该上限  $R$  即称为 HNN 的相对记忆容量  $M_0 = NR$  称为绝对记忆容量。当记忆项的个数  $M \leq M_0$  时, 式 (4-165) 成立, 即每个记忆项都是吸引子; 反之, 当  $M > M_0$  时式 (4-165) 不成立。按照 Hopfield 的论点, HNN 的相对容量是  $R = 0.15$ , 但是后来的模拟实验和理论分析结论都否定了此论点。R. J. McEliece 等的经典论文<sup>[68]</sup> 采用概率和统计的方法求得了当  $N \rightarrow \infty$  时的  $R$  渐近值为

$$\lim_{N \rightarrow \infty} R = \frac{1}{2 \ln N} \quad (4-166)$$

由于有关的理论分析较繁, 此处不给证明, 有兴趣的读者可参阅文献<sup>[68]</sup> 或文献<sup>[67]</sup>。  $1/(2 \ln N)$  这个相对记忆容量虽然是一个渐近值, 但是各种模拟实验证实, 当  $N \geq 64$  时实际的容量与此理论计算值已相当接近<sup>[67]</sup>。可以看到, 采用标准的 HNN 作为联想记忆器时, 其记忆效率即容量是相当低的。例如 当  $N = 1000$  时  $R = 0.0724$  相应的绝对记忆容量是  $M_0 = 72$ 。而且随着  $N$  的增大  $R$  值将不断降低。

第二个问题涉及吸引域的问题, 为此须先给出若干定义。首先给出两个状态之间的汉明距离的定义。设有  $X^a, X^b$  两个状态, 它们之间的相对汉明距离定义为 (下面简称为距离)

$$d(\mathbf{X}^a, \mathbf{X}^b) = \frac{1}{2N} \sum_{i=1}^N |x_i^a - x_i^b| \quad (4-167)$$

若  $\mathbf{X}^a$  和  $\mathbf{X}^b$  完全相同, 则其距离为 0; 若只有一位不同, 则距离为  $1/N$  等等。若完全不同则距离为 1。现在, 当一个 HNN 的记忆项个数设置为  $M \leq M_0 = RN$  而且按同步方式 (式 (4-158)) 或非同步方式 (式 (4-161)) 提取记忆时, 可以考察用缺损的样本  $\mathbf{v}^{(m)}$  作为初始状态  $\mathbf{X}(0)$  能否提取出  $\mathbf{X}^{(m)}$  即系统能否收敛到  $\mathbf{X}^{(m)}$ 。如果  $\mathbf{X}^{(m)}$  与  $\mathbf{X}^{(m)}$  之间的距离为 0 则无论采用哪一种方式都能实现正确提取。如二者之间的距离超出某个上限  $D$  之后就不能再实现正确提取, 那么可以将每个记忆项  $\mathbf{X}^{(m)}$  为中心且半径为  $D$  的超球定义为“吸引域”,  $D$  称为吸引半径。对于同步运行方式的一步提取和多步提取以及非同步方式的提取都应确定其  $D$  值与记忆项个数  $M$  之间的关系。文献[68] 也给出了上列这些问题的答案, 其结论是 (此结论是  $N \rightarrow \infty$  时的渐近值):

(1) 若按同步方式实现一步提取, 那么当存储项的个数  $M$  与吸引半径  $D$  之间有下列关系时:

$$M = (1 - 2D)^2 \frac{N}{2 \ln N} = (1 - 2D)^2 M_0, \quad 0 \leq D < \frac{1}{2} \quad (4-168)$$

实现正确提取的概率为 1。其严格证明可查阅文献[68] 或文献[67]。可以看到, 若  $D = 0$  则  $M$  达到一个标准 HNN 绝对记忆容量值  $M_0$  即  $M = M_0 = NR = N/(2 \ln N)$ 。若要求  $D$  有较大值, 则  $M$  必须相应降低,  $D$  不允许达到或超过  $1/2$ 。当  $D = \frac{1}{2}$  时,  $M = 0$ , 这时不能记忆任何存储项。

(2) 若按同步方式实现二步提取, 那么当  $M$  和  $D$  同时满足下列各式时, 实现正确提取的概率为 1 (证明见文献[68] 或文献[67]):

$$M \leq (1 - \delta) \frac{N}{2 \ln N}, \quad 0 < \delta \ll 1; \quad d \leq D = \frac{1}{2} - \epsilon, \quad 0 < \epsilon \ll 1 \quad (4-169)$$

可以看到, 只要存储项的个数  $M$  略小于绝对记忆容量  $M_0 = NR = N/(2 \ln N)$  那么二步提取的吸引半径只是略小于  $1/2$ 。

(3) 在非同步运行的情况下实现多步提取, 其  $M$  和  $D$  满足式 (4-169) 时, 实现正确提取的概率为 1。此结果与同步运行二步提取一致。

应该指出, 以上结论都是  $N \rightarrow \infty$  时的渐近结果。模拟计算表明, 当  $N \geq 256$  时所得到的无误提取  $M$  和  $D$  值与理论计算值已非常接近。当  $N < 256$  时,  $M$  和  $D$  的实际取值均需较理论计算值偏小才能实现正确提取,  $N$  值越小, 向下修正的幅度越大。

文献[68] 给出的结论是在各记忆项和其中各分量均相互独立并且各分量取 1 和 -1 的概率相同的前提下得到的。而实际情况下各记忆项往往存在相关性, 这时实现无误提取的  $M$  和  $D$  值都要低于上面给出的结果而且取决于是何种相关性。此外, 还存在着这样一些问题: 吸引域可能不是纯圆形的超球 (即各维的半径相同) 而是椭圆形超球。在不同记忆项处的吸引域大小不一致。存在虚假 (spurious) 吸引子, 这些吸引子并不是记忆项。对于这些问题只用概率和统计方法就不够了。在 4.5.3 小节和 4.5.4 小节将介绍几种分析和解决这些问题的方法。



#### 4.5.2 通过改变标准 HNN 的神经元输入输出函数来加大记忆容量和扩大吸引域

标准 HNN 的一个大缺点是存储效率甚低, 即其相对记忆容量值  $R$  仅为  $1/(2\ln N)$  而且当装载率  $r$  接近  $R$  时吸引半径  $D$  趋于 0。本节讨论的改进思路是保持按照 Hebb 学习算法构成的权矩阵  $\mathbf{W}$  不变 (式 4-164), 而通过改变神经元函数来扩大  $R$  和  $D$ 。文献 [69] 建议将式 (4-158) 给出的神经元函数进行修改。首先, 该式可写成

$$\mathbf{X}(t+1) = \text{sgn}[\mathbf{W}\mathbf{X}(t)] \quad (4-170)$$

其中  $\text{sgn}[\cdot]$  表示: 若  $\mathbf{U} = [u_1, u_2, \dots, u_N]^T$ , 则  $\text{sgn}[\mathbf{U}] = [\text{sgn}[u_1], \text{sgn}[u_2], \dots, \text{sgn}[u_N]]^T$ 。修改后则成为

$$\mathbf{X}(t+1) = \text{sgn}\{\mathbf{W}[\mathbf{X}(t) + \Phi(\mathbf{W}\mathbf{X}(t))]\} \quad (4-171)$$

其中  $\Phi(\mathbf{U}) = [\varphi(u_1), \varphi(u_2), \dots, \varphi(u_N)]^T$  且

$$\varphi(u_i) = \begin{cases} C, & u_i < -h \\ 0, & -h \leq u_i \leq h \\ -C, & u_i > h \end{cases} \quad (4-172)$$

$C \approx 2.7, h \approx 1.9$ 。文献 [70] 在此基础上改用如下  $\varphi(\cdot)$  函数:

$$\varphi(u_i) = -au_i + C\text{sgn}[u_i] \quad (4-173)$$

$a$  和  $c$  是两个可调节的参数。本节只介绍基于此种修正的 HNN 的记忆容量和吸引域, 下面分记忆容量和吸引域两部分对此修正 HNN 进行分析。此分析只讨论记忆的一步提取 (同步运行情况) 分析基于  $N \rightarrow \infty$  时可采用概率统计中的渐近方法, 从而使数学计算成为可能。

##### 1. 记忆容量的计算

设有  $M$  个记忆项  $\mathbf{X}^{(m)}, m = 1 \sim M$ 。HNN 的权矩阵  $\mathbf{W}$  用式 (4-164) 的第 1 个公式计算。现在要从某个特定记忆项  $\mathbf{X}^{(\mu)}$  的缺损样本  $\mathbf{X}^{(\mu)}$  来提取出  $\mathbf{X}^{(\mu)}$  为此令 HNN 的初始状态为  $\mathbf{X}(0) = \mathbf{X}^{(\mu)}$  若经过一步运行使得  $\mathbf{X}(1) = \mathbf{X}^{(\mu)}$  则实现了正确提取。为简化书写, 下面用  $\mathbf{X}$  来表示  $\mathbf{X}(0)$ , 用  $\tilde{\mathbf{X}}$  表示  $\mathbf{X}(1)$ , 用  $d$  表示  $\mathbf{X}^{(\mu)}$  与  $\mathbf{X}$  之间的相对汉明距离 (式 4-167) 用  $\tilde{d}$  表示  $\mathbf{X}^{(\mu)}$  与  $\tilde{\mathbf{X}}$  之间的相对汉明距离。当  $N \rightarrow \infty$  时  $\tilde{d}$  是一个具有正态分布的随机变量, 如果其统计平均值  $E[\tilde{d}] = \langle \tilde{d} \rangle < 1/N$  则表明  $\tilde{\mathbf{X}}$  与  $\mathbf{X}^{(\mu)}$  完全一致, 即实现了正确提取。下面先用标准 HNN 为例来进行此项分析。

由式 (4-170) 设  $\mathbf{U} = \mathbf{W}\mathbf{X}$  则  $\tilde{\mathbf{X}} = \text{sgn}[\mathbf{U}]$ 。  $\mathbf{U}$  的各分量  $u_i, i = 1 \sim N$  可求得如下。利用式 (4-164) 第 1 式得到:

$$\left. \begin{aligned} \mathbf{U} = \mathbf{W}\mathbf{X} &= \frac{1}{N} \sum_{m=1}^M \mathbf{X}^{(m)} (\mathbf{X}^{(m)})^T \mathbf{X} - \frac{M}{N} \mathbf{X} = \sum_{m=1}^M \mathbf{X}^{(m)} \varphi^{(m)} - \frac{M}{N} \mathbf{X} \\ \varphi^{(m)} &= \frac{1}{N} (\mathbf{X}^{(m)})^T \mathbf{X} = \frac{1}{N} \sum_{j=1}^N x_j^{(m)} x_j \end{aligned} \right\} \quad (4-174)$$

这样可得:

$$u_i = \sum_{m=1}^M x_i^{(m)} \varphi^{(m)} - \frac{M}{N} x_i, \quad i = 1 \sim N \quad (4-175)$$

设

$$L_i^{(m)} = \frac{1}{N} \sum_{\substack{j=1 \\ j \neq i}}^N x_j^{(m)} x_j, \quad \varphi^{(m)} = L_i^{(m)} + \frac{1}{N} x_i^{(m)} x_i \quad (4-176)$$

将式 (4-176) 代入式 (4-175) 即得

$$u_i = \sum_{m=1}^M x_i^{(m)} L_i^{(m)}, \quad i = 1 \sim N \quad (4-177)$$

将此式中与待提取项  $\mathbf{X}^{(\mu)}$  有关部分和无关部分写成两部分：

$$u_i = x_i^{(\mu)} L_i^{(\mu)} + \sum_{\substack{m=1 \\ m \neq \mu}}^M x_i^{(m)} L_i^{(m)} = x_i^{(\mu)} L_i^{(\mu)} + \frac{1}{N} \sum_{\substack{m=1 \\ m \neq \mu}}^M \sum_{\substack{j=1 \\ j \neq i}}^N x_i^{(m)} x_j^{(m)} x_j$$

由于  $x_i^{(m)}, x_j^{(m)}$  与  $x_j$  独立无关，所以各  $x_i^{(m)} x_j^{(m)} x_j$  是相互独立随机变量，其取  $-1$  和  $1$  的概率都是  $1/2$ 。根据大数定律和中心极限定理<sup>[71]</sup> 当  $N \rightarrow \infty$  时上式最右侧第 2 项将趋于一个正态随机变量，其均值为 0 其均方差值为  $\sigma^2 = (M-1)(N-1)/N^2$ 。当  $N$  和  $M$  均很大时  $\sigma^2 \approx M/N = r$ 。这样该式可以写成

$$u_i = x_i^{(\mu)} L_i^{(\mu)} + \sigma \epsilon_{\mathcal{M}_i}, \quad i = 1 \sim N \quad (4-178)$$

其中  $\sigma = \sqrt{M/N} = \sqrt{r}$ ， $\epsilon_{\mathcal{M}_i}$  是一个均值为 0 均方差值为 1 的正态随机变量，可以用  $\mathcal{N}(0,1)$  表示其概率密度函数  $\theta(t)$  计算如下：

$$\theta(t) = \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} \quad (4-179)$$

对于本节讨论的修正 HNN 由式 (4-171) 设  $\mathbf{U} = \mathbf{W}\mathbf{X} + \mathbf{W}\Phi(\mathbf{W}\mathbf{X})$ ,  $\tilde{\mathbf{X}} = \text{sgn}[\mathbf{U}]$  其中  $\Phi(\cdot)$  由式 (4-173) 确定。那么依照上述标准 HNN 的推导过程，可以求得  $\mathbf{U}$  的各分量为

$$u_i = x_i^{(\mu)} L_i^{(\mu)} + Bx_i + \sigma \epsilon_{\mathcal{M}_i} \quad (4-180)$$

其中  $\epsilon_{\mathcal{M}_i}$  仍是一个  $\mathcal{N}(0,1)$  随机变量。式中的  $L_i^{(\mu)}$ ,  $B$  和  $\sigma$  诸参数值及此式的推导将稍后给出。式 (4-178) 和式 (4-180) 可以归并成一个如下式子：

$$u_i = x_i^{(\mu)} (L_i^{(\mu)} + Bx_i^{(\mu)} x_i + \sigma \epsilon_{\mathcal{M}_i}) \quad (4-181)$$

对于标准 HNN,  $B = 0$  对于修正 HNN,  $B \neq 0$  而且二者的  $L_i^{(\mu)}$  和  $\sigma$  值也不相同。这样对于二者都可以求得

$$\tilde{x}_i = \text{sgn}[u_i] = \text{sgn}[x_i^{(\mu)} (L_i^{(\mu)} + Bx_i^{(\mu)} x_i + \sigma \epsilon_{\mathcal{M}_i})], \quad i = 1 \sim N \quad (4-182)$$

$\tilde{\mathbf{X}} = \mathbf{X}(1)$  与  $\mathbf{X}^{(\mu)}$  之间的相对汉明距离  $\tilde{d}$  可以求得如下：

$$\tilde{d} = \frac{1}{2} \left[ 1 - \frac{1}{N} \sum_{i=1}^N x_i^{(\mu)} \tilde{x}_i \right] = \frac{1}{2} [1 - l(\mathbf{X}^{(\mu)}, \tilde{\mathbf{X}})] \quad (4-183)$$

其中  $l(\mathbf{X}^{(\mu)}, \tilde{\mathbf{X}})$  称为  $\mathbf{X}^{(\mu)}$  与  $\tilde{\mathbf{X}}$  之间的“交迭”(overlap) 若二者完全一致 则交迭为 1 若完全不一致，则交迭为  $-1$ 。根据式 (4-182) 可求得

$$l(\mathbf{X}^{(\mu)}, \tilde{\mathbf{X}}) = \frac{1}{N} \sum_{i=1}^N x_i^{(\mu)} \tilde{x}_i = \frac{1}{N} \sum_{i=1}^N \text{sgn}[L_i^{(\mu)} + Bx_i^{(\mu)} x_i + \sigma \epsilon_{\mathcal{M}_i}] \quad (4-184)$$

若  $\mathbf{X}^{(\mu)}$  与  $\mathbf{X}$  之间的相对汉明距离为  $d$  那么  $x_i^{(\mu)}$  与  $x_i$  相一致的有  $N(1-d)$  项 不一致的有  $Nd$  项 前者  $x_i^{(\mu)} x_i = 1$  后者  $x_i^{(\mu)} x_i = -1$ 。如果将前者的集合记为  $\Omega^+$  后者的集合记为  $\Omega^-$ ，那么式 (4-184) 可以写成

$$l(\mathbf{X}^{(\mu)}, \tilde{\mathbf{X}}) = \frac{1}{N} \sum_{\substack{i=1 \\ i \in \hat{\mathcal{O}}^+}}^N \text{sgn}[L_i^{(\mu)} + B + \sigma \epsilon_{\mathcal{M}_i}] + \frac{1}{N} \sum_{\substack{i=1 \\ i \in \hat{\mathcal{O}}^-}}^N \text{sgn}[L_i^{(\mu)} - B + \sigma \epsilon_{\mathcal{M}_i}]$$

对于标准 HNN 由式 (4-176) 可知 当  $N \rightarrow \infty$  时  $L_i^{(\mu)} \rightarrow l^{(\mu)}$  即

$$\lim_{N \rightarrow \infty} L_i^{(\mu)} = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{\substack{j=1 \\ j \neq i}}^N x_j^{(\mu)} x_j = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{j=1}^N x_j^{(\mu)} x_j = \lim_{N \rightarrow \infty} l^{(\mu)}$$

这样当  $N \rightarrow \infty$  时, 下列渐近公式成立:

$$l(\mathbf{X}^{(\mu)}, \tilde{\mathbf{X}}) = (1-d) \text{sgn}[l^{(\mu)} + B + \sigma \epsilon_{\mathcal{M}_i}] + d \text{sgn}[l^{(\mu)} - B + \sigma \epsilon_{\mathcal{M}_i}] \quad (4-185)$$

其中  $l^{(\mu)}$  为  $\mathbf{X}^{(\mu)}$  与  $\mathbf{X}$  之间的交迭, 即

$$l^{(\mu)} = \frac{1}{N} \sum_{j=1}^N x_j^{(\mu)} x_j \quad (4-186)$$

这样, 由式 (4-183) 和式 (4-185) 可以求得

$$\langle \tilde{d} \rangle = \frac{1}{2} [1 - \langle l(\mathbf{X}^{(\mu)}, \mathbf{X}) \rangle]$$

$$\langle l(\mathbf{X}^{(\mu)}, \mathbf{X}) \rangle = (1-d) \langle \text{sgn}[l^{(\mu)} + B + \sigma \epsilon_{\mathcal{M}_i}] \rangle + d \langle \text{sgn}[l^{(\mu)} - B + \sigma \epsilon_{\mathcal{M}_i}] \rangle$$

易于求得

$$\langle \text{sgn}[l^{(\mu)} + B + \sigma \epsilon_{\mathcal{M}_i}] \rangle = 1 - 2Pr(l^{(\mu)} + B + \sigma \epsilon_{\mathcal{M}_i} < 0)$$

$$\langle \text{sgn}[l^{(\mu)} - B + \sigma \epsilon_{\mathcal{M}_i}] \rangle = 1 - 2Pr(l^{(\mu)} - B + \sigma \epsilon_{\mathcal{M}_i} < 0)$$

且

$$Pr(l^{(\mu)} + B + \sigma \epsilon_{\mathcal{M}_i} < 0) = \psi\left(\frac{l^{(\mu)} + B}{\sigma}\right)$$

$$Pr(l^{(\mu)} - B + \sigma \epsilon_{\mathcal{M}_i} < 0) = \psi\left(\frac{l^{(\mu)} - B}{\sigma}\right)$$

其中

$$\psi(u) = \int_{-\infty}^{-u} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt \quad (4-187)$$

因此

$$\langle \tilde{d} \rangle = (1-d) \psi\left(\frac{l^{(\mu)} + B}{\sigma}\right) + d \psi\left(\frac{l^{(\mu)} - B}{\sigma}\right) \quad (4-188)$$

对于修正 HNN, 下文也将证明 当  $N \rightarrow \infty$  时,  $L_i^{(\mu)}$  与  $i$  无关 可以用  $L^{(\mu)}$  表示 这无论是标准 HNN 还是修正 HNN,  $\langle \tilde{d} \rangle$  可用下列统一公式计算:

$$\langle \tilde{d} \rangle = (1-d) \psi\left(\frac{L^{(\mu)} + B}{\sigma}\right) + d \psi\left(\frac{L^{(\mu)} - B}{\sigma}\right) \quad (4-189)$$

对于标准 HNN,  $L^{(\mu)} = l^{(\mu)}$ ,  $\sigma = \sqrt{r}$ ,  $B = 0$  对于修正 HNN 其  $L^{(\mu)}$ ,  $B$ ,  $\sigma$  的计算在紧接下文给出。

依据式 (4-171) 在修正 HNN 的情况下,

$$\tilde{\mathbf{X}} = \text{sgn}[\mathbf{U}], \quad \mathbf{U} = \mathbf{W}\mathbf{X} + \mathbf{W}\Phi(\mathbf{W}\mathbf{X}) \quad (4-190)$$

$\mathbf{U}$  的第  $i$  分量  $u_i$  可表为  $(\mathbf{W}\mathbf{X})$  第  $i$  分量  $(\mathbf{W}\mathbf{X})_i$  及  $(\mathbf{W}\Phi(\mathbf{W}\mathbf{X}))$  第  $i$  分量  $(\mathbf{W}\Phi(\mathbf{W}\mathbf{X}))_i$  之和 即

$$u_i = (\mathbf{W}\mathbf{X})_i + (\mathbf{W}\Phi(\mathbf{W}\mathbf{X}))_i, \quad i = 1 \sim N \quad (4-191)$$

据式(4-178)和式(4-186)当  $N \rightarrow \infty$  时成立

$$(\mathbf{WX})_i = x_i^{(\mu)} l^{(\mu)} + \sqrt{r} \epsilon_{\mathcal{N}_i} \quad (4-192)$$

其中  $l^{(\mu)}$  是  $\mathbf{X}^{(\mu)}$  与  $\mathbf{X}$  之交迭(见式(4-186)),  $\epsilon_{\mathcal{N}_i}$  计算公式为(参见式(4-178)的说明)

$$\epsilon_{\mathcal{N}_i} = \frac{1}{\sqrt{r}} \left( \frac{1}{N} \sum_{\substack{p=1 \\ p \neq \mu}}^M \sum_{\substack{k=1 \\ k \neq i}}^N x_i^{(p)} x_k^{(p)} x_k \right) \quad (4-193)$$

且已知当  $N \rightarrow \infty$  时  $\epsilon_{\mathcal{N}_i} \rightarrow \mathcal{N}(0, 1)$ 。

对于式(4-191)右侧第2项,据式(4-164)并引用式(4-192)即得

$$(\mathbf{W}\Phi(\mathbf{WX}))_i = \frac{1}{N} \sum_{m=1}^M \sum_{\substack{j=1 \\ j \neq i}}^N x_i^{(m)} x_j^{(m)} \varphi(x_j^{(\mu)} l^{(\mu)} + \sqrt{r} \epsilon_{\mathcal{N}_j}) \quad (4-194)$$

其中函数  $\varphi(\cdot)$  由式(4-173)给出。仍据中心极限定理,可知  $\mathbf{W}\Phi(\mathbf{WX})_i$  当  $N \rightarrow \infty$  时是正态随机变量。这样,  $u_i$  是正态随机变量,只须确定其均值  $\langle u_i \rangle$  和均方差值  $\sigma^2$  即可确定其概率分布。由式(4-191)可得

$$\langle u_i \rangle = \langle (\mathbf{WX})_i \rangle + \langle (\mathbf{W}\Phi(\mathbf{WX}))_i \rangle \quad (4-195)$$

由式(4-192)

$$\langle (\mathbf{WX})_i \rangle = x_i^{(\mu)} l^{(\mu)} \quad (4-196)$$

为计算  $\langle (\mathbf{W}\Phi(\mathbf{WX}))_i \rangle$  首先注意  $\varphi(\cdot)$  是奇对称函数,因此

$$\varphi(x_j^{(\mu)} l^{(\mu)} + \sqrt{r} \epsilon_{\mathcal{N}_j}) = \varphi(x_j^{(\mu)} (l^{(\mu)} + \sqrt{r} \tilde{\epsilon}_{\mathcal{N}_j})) = x_j^{(\mu)} \varphi(l^{(\mu)} + \sqrt{r} \tilde{\epsilon}_{\mathcal{N}_j}) \quad (4-197)$$

其中  $\tilde{\epsilon}_{\mathcal{N}_j} = x_j^{(\mu)} \epsilon_{\mathcal{N}_j}$  当  $N \rightarrow \infty$  时,也是一个  $\mathcal{N}(0, 1)$  随机变量。其次,将式(4-194)右侧分成两项并代入式(4-197)得到

$$(\mathbf{W}\Phi(\mathbf{WX}))_i = \frac{1}{N} \sum_{\substack{j=1 \\ j \neq i}}^N x_i^{(\mu)} \varphi(l^{(\mu)} + \sqrt{r} \tilde{\epsilon}_{\mathcal{N}_j}) + \frac{1}{N} \sum_{\substack{m=1 \\ m \neq \mu}}^M \sum_{\substack{j=1 \\ j \neq i}}^N x_i^{(m)} x_j^{(m)} x_j^{(\mu)} \varphi(l^{(\mu)} + \sqrt{r} \tilde{\epsilon}_{\mathcal{N}_j}) \quad (4-198)$$

这样,此式左侧的均值等于其右侧二项各自均值之和。其中第一项的均值为

$$\frac{x_i^{(\mu)}}{N} \sum_{\substack{j=1 \\ j \neq i}}^N \langle \varphi(l^{(\mu)} + \sqrt{r} \tilde{\epsilon}_{\mathcal{N}_j}) \rangle$$

由于  $\tilde{\epsilon}_{\mathcal{N}_j}$  是一个  $\mathcal{N}(0, 1)$  随机变量,则易于求得

$$\langle \varphi(l^{(\mu)} + \sqrt{r} \tilde{\epsilon}_{\mathcal{N}_j}) \rangle = \int_{-\infty}^{+\infty} \varphi(l^{(\mu)} + \sqrt{r} t) \cdot \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt = F_1(l^{(\mu)}) \quad (4-199)$$

当  $N \rightarrow \infty$  时,第一项的均值即逼近于

$$x_i^{(\mu)} F_1(l^{(\mu)})$$

第二项的均值为

$$\frac{1}{N} \sum_{\substack{m=1 \\ m \neq \mu}}^M \sum_{\substack{j=1 \\ j \neq i}}^N \langle x_i^{(m)} x_j^{(m)} x_j^{(\mu)} \varphi(l^{(\mu)} + \sqrt{r} \tilde{\epsilon}_{\mathcal{N}_j}) \rangle$$

借助式(4-193),  $\sqrt{r} \tilde{\epsilon}_{\mathcal{N}_j}$  可表示为

$$\sqrt{r} \tilde{\epsilon}_{\mathcal{N}_j} = \frac{1}{N} x_j^{(m)} x_i^{(m)} x_j^{(\mu)} x_i + \frac{1}{N} \sum_{\substack{p=1 \\ p \neq m, \mu}}^M \sum_{\substack{k=1 \\ k \neq j, i}}^N x_j^{(p)} x_k^{(p)} x_j^{(\mu)} x_k + \frac{1}{N} \sum_{\substack{p=1 \\ p \neq m, \mu}}^M x_j^{(p)} x_i^{(p)} x_j^{(\mu)} x_i$$

注意 当  $N \rightarrow \infty$  时 此式右侧第 2、3 项之和逼近一个正态随机变量，其均值为 0 均方差为  $\sqrt{r}$  且与右侧第 1 项无关，可以表示为  $\sqrt{r} \hat{\epsilon}_{\mathcal{N}_j} \circ \hat{\epsilon}_{\mathcal{N}_j}$  为  $\mathcal{N}(0, 1)$  随机变量且与  $x_j^{(m)} x_i^{(m)} x_j^{(\mu)}$  无关。设  $x_j^{(m)} x_i^{(m)} x_j^{(\mu)} = \alpha^{(m)}$  这是一个取 1 和 -1 的概率皆等于 0.5 的随机变量。这样 第 2 项的均值可用下式计算：

$$\frac{1}{N} \sum_{\substack{m=1 \\ m \neq \mu}}^M \sum_{\substack{j=1 \\ j \neq i}}^N \langle \alpha^{(m)} \varphi(l^{(\mu)} + \frac{x_i}{N} \alpha^{(m)} + \sqrt{r} \hat{\epsilon}_{\mathcal{N}_j}) \rangle \quad (4-200)$$

直接求此和式内的各统计平均值很难，故取一种求解的近似方法：先令  $\alpha^{(m)}$  不变 对  $\hat{\epsilon}_{\mathcal{N}_j}$  取平均 再对  $\alpha^{(m)}$  取平均。第一步所求结果是 (见式 4-199))

$$\alpha^{(m)} \langle \varphi(l^{(\mu)} + \frac{x_i}{N} \alpha^{(m)} + \sqrt{r} \hat{\epsilon}_{\mathcal{N}_j}) \rangle = \alpha^{(m)} F_1 \left( l^{(\mu)} + \frac{x_i}{N} \alpha^{(m)} \right)$$

此函数用下列函数来近似计算：

$$F_1 \left( l^{(\mu)} + \frac{x_i}{N} \alpha^{(m)} \right) = F_1(l^{(\mu)}) + \dot{F}_1(l^{(\mu)}) \frac{x_i}{N} \alpha^{(m)}$$

其中  $F_1(l^{(\mu)}) = dF_1(u)/du|_{u=l^{(\mu)}}$ 。这样可做第二步 对  $\alpha^{(m)}$  平均，结果如下：

$$\langle \alpha^{(m)} F_1(l^{(\mu)}) \rangle + \langle \dot{F}_1(l^{(\mu)}) \frac{x_i}{N} (\alpha^{(m)})^2 \rangle$$

其中第一个平均值为 0。第二个平均值为  $\frac{x_i}{N} \dot{F}_1(l^{(\mu)})$ 。当  $M, N$  足够大时，式 4-200 的渐近

值是  $\frac{M}{N} x_i \dot{F}_1(l^{(\mu)})$  因此

$$(\langle \mathbf{W} \Phi(\mathbf{W} \mathbf{X}) \rangle)_i = x_i^{(\mu)} F_1(l^{(\mu)}) + r x_i F_1(l^{(\mu)}) \quad (4-201)$$

将式 (4-201) 和式 (4-196) 代入式 (4-195) 即得

$$\langle u_i \rangle = x_i^{(\mu)} (l^{(\mu)} + F_1(l^{(\mu)})) + r x_i F_1(l^{(\mu)}) \quad (4-202)$$

$u_i$  的均方差  $\sigma^2$  也可按此思路求得，但因计算过程繁琐此处不再推导，只将其值列于下面的式 (4-203) 中 (推导计算见文献[70])。这样 修正 HNN 关于  $u_i$  表达式的式 (4-180) 中所涉及各个参数可列出如下：

$$\left. \begin{aligned} L_i^{(\mu)} &= l^{(\mu)} + F_1(l^{(\mu)}) \\ B &= r \dot{F}_1(l^{(\mu)}) \\ \sigma^2 &= r \{ 1 + F_2(l^{(\mu)}) + 2 l^{(\mu)} F_1(l^{(\mu)}) \dot{F}_1(l^{(\mu)}) + (\dot{F}_1(l^{(\mu)}))^2 + \\ &\quad 2 [l^{(\mu)} F_1(l^{(\mu)}) + \dot{F}_1(l^{(\mu)})] \} \\ \epsilon_{\mathcal{N}_i} &\in \mathcal{N}[0, 1] \end{aligned} \right\} \quad (4-203)$$

其中  $l^{(\mu)}$  用式 (4-186) 计算， $F_1(l^{(\mu)})$  和  $\dot{F}_1(l^{(\mu)})$  用式 (4-199) 计算， $r = M/N$ ， $F_2(l^{(\mu)})$  用下式计算：

$$F_2(l^{(\mu)}) = \int_{-\infty}^{+\infty} \varphi^2(l^{(\mu)} + \sqrt{r} t) \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt \quad (4-204)$$

现在即可用此结果来计算修正 HNN 的记忆容量。先讨论一个较简单的情况：式 (4-173)

中的参数  $C = 0$  这时函数  $\varphi(\cdot)$  为

$$\varphi(u) = -au \quad (4-205)$$

将此式代入式 (4-199) 和式 (4-205) 即可求得

$$F_1(l^{(\mu)}) = -al^{(\mu)}, \quad F_1(l^{(\mu)}) = -a, \quad F_2(l^{(\mu)}) = a^2((l^{(\mu)})^2 + r)$$

这样由式 (4-203) 可求得

$$L_i^{(\mu)} = l^{(\mu)}(1-a), \quad B = -ar, \quad \sigma^2 = r\{1 + a^2[1 + r + 3(l^{(\mu)})^2] - 2a[1 + (l^{(\mu)})^2]\}$$

由式 (4-189) 可以计算出, 以输入缺损样本  $\mathbf{X}^{(\mu)}$  作为  $t = 0$  时刻系统的状态  $\mathbf{X}(0) = \mathbf{X}$  即有  $\mathbf{X} = \mathbf{X}(0) = \hat{\mathbf{X}}^{(\mu)}$  且  $\mathbf{X}$  与  $\mathbf{X}^{(\mu)}$  之间的交迭为  $l^{(\mu)}$  时,  $t = 1$  时刻系统的状态  $\mathbf{X}(1) = \tilde{\mathbf{X}}$  与  $\mathbf{X}^{(\mu)}$  之间相对汉明距离  $\tilde{d}$  的统计平均值为

$$\langle \tilde{d} \rangle = (1-d)\psi\left(\frac{l^{(\mu)}(1-a) - ar}{\sigma}\right) + d\psi\left(\frac{l^{(\mu)}(1-a) + ar}{\sigma}\right) \quad (4-206)$$

其中  $d = \frac{1}{2}(1 - l^{(\mu)})$ 。

在计算记忆容量时令  $\mathbf{X}^{(\mu)} = \mathbf{X}^{(\mu)}$  相应  $l^{(\mu)} = 1$  即有  $d = 0$  这样可以求得

$$\langle \tilde{d} \rangle = \psi(G), \quad G = \frac{1 - a(1+r)}{\{r[(1-2a)^2 + ra^2]\}^{\frac{1}{2}}} \quad (4-207)$$

图 4-9 描绘了当  $N = 500$  且装载率  $r$  取 0.2, 0.3, 0.4 三种不同值时, 由计算机模拟所得  $\langle \tilde{d} \rangle$  随  $a$  而变化的曲线。此结果是通过 10 次提取所得  $\tilde{d}$  取平均而得到。由这些曲线可以看到, 在  $a = 0.5$  附近  $\langle \tilde{d} \rangle$  达到其最小值。 $r$  值越小,  $\langle \tilde{d} \rangle \sim a$  曲线的最小值点越接近  $a = 0.5$ 。实

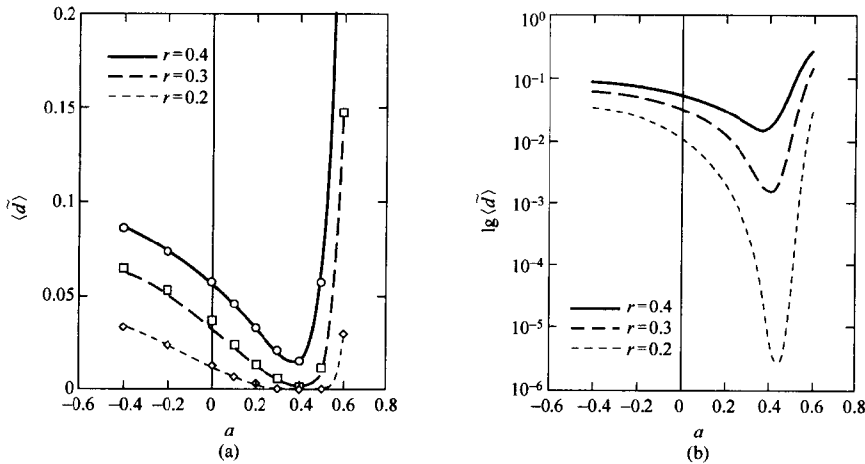


图 4-9  $\langle \tilde{d} \rangle \sim a$  曲线

际上由式 (4-207) 可见,  $G$  越大则  $\langle \tilde{d} \rangle$  越小。当  $r$  值一定时, 由  $dG/da = 0$  即可求得使  $G$  达到最大, 从而使  $\langle \tilde{d} \rangle$  达到最小的  $a$  值是

$$a = (1-r)/(2-r) \quad (4-208)$$

若  $r$  足够小, 则  $a \approx 0.5$ 。由式 (4-207) 还可以看到, 当  $a$  值一定时,  $\langle \tilde{d} \rangle$  随  $r$  的减小而单调下降。这样就能用下列准则来界定一个 HNN 的记忆容量: 若  $r$  增加到超过某个上界值  $R$

时,  $\langle \tilde{d} \rangle \leq 1/N$  的条件不再成立 那么  $R$  就是该 HNN 的相对记忆容量,  $M_0 = NR$  即为其绝对记忆容量。此条件可以写成

$$N\langle \tilde{d} \rangle = N\psi(G) = 1, \quad G = \frac{1 - a(1+R)}{\{R[(1-2a)^2 + Ra^2]\}^{\frac{1}{2}}} \quad (4-209)$$

当  $R$  足够小从而  $G$  足够大时 ( 实际应用中都是这种情况 ),  $\psi(G)$  可用下列渐近公式计算:

$$\psi(G) \approx \theta(G)/G \quad (4-210)$$

其中  $\theta(G)$  见式 4-179)。对式 (4-209) 取对数, 且代入式 (4-210) 和 (4-179) 即得到

$$\ln[N\psi(G)] = \ln N - \ln G - \frac{1}{2} \ln 2\pi - \frac{G^2}{2} = 0$$

当  $N$  和  $G$  足够大时,  $-\ln G - \frac{1}{2} \ln 2\pi$  与它们相比可以略之不计, 则此条件可以写成

$$G^2 = \frac{[1 - a(1+R)]^2}{R[(1-2a)^2 + Ra^2]} = 2\ln N \quad (4-211)$$

对于标准 HNN,  $a = 0$  则  $G^2 = R^{-1}$ , 则由此式求得其相对和绝对记忆容量  $R, M_0$  分别为

$$R = \frac{1}{2\ln N}, \quad M_0 = \frac{N}{2\ln N} \quad (4-212)$$

此结果与 4.5.1 小节介绍的文献[68]用更精确方法导出的结果一致 ( 见式 (4-168))。

对于修正 HNN 且取最优参数值  $a = 0.5$  则可求得  $G^2 = [(1/R) - 1]^2$  当  $R$  足够小时  $G^2 \approx R^{-2}$ , 这样就能求得与之相应的  $R$  和  $M_0$  为

$$R = \frac{1}{\sqrt{2\ln N}}, \quad M_0 = \frac{N}{\sqrt{2\ln N}} \quad (4-213)$$

对比以上两种结果, 可以看到取最优  $a$  值的修正 HNN 的记忆容量较标准 HNN 有明显提高。例如 当  $N = 1\,000$  时, 后者的  $R = 0.072, M_0 = 72$ ; 前者的  $R = 0.269, M_0 = 269$ 。前者的记忆容量约为后者的 3.7 倍 且此差距随  $N$  的增加而进一步扩大。

其次 讨论式 (4-173) 中函数  $\varphi(\cdot)$  的参数  $a$  和  $C$  取任意值的情况。为求  $\langle \tilde{d} \rangle$  仍须求  $L^{(\mu)}$  和  $\sigma^2$ 。这些变量仍可用式 4-203 计算 为此需求得  $F_1(l^{(\mu)}), \dot{F}_1(l^{(\mu)}), F_2(l^{(\mu)})$ 。分别用式 (4-199) 和式 4-204) 将  $\varphi(u) = -au + C\text{sgn}[u], u = l^{(\mu)} + \sqrt{r}t$  代入 即得到

$$\begin{aligned} F_1(l^{(\mu)}) &= \int_{-\infty}^{-l^{(\mu)}/\sqrt{r}} [-a(l^{(\mu)} + \sqrt{r}t) - C] \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt + \\ &\quad \int_{-l^{(\mu)}/\sqrt{r}}^{\infty} [-a(l^{(\mu)} + \sqrt{r}t) + C] \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt \\ &= -al^{(\mu)} + C[1 - 2\psi(l^{(\mu)}/\sqrt{r})] \end{aligned} \quad (4-214)$$

$$\dot{F}_1(l^{(\mu)}) = -a - 2C\dot{\psi}(l^{(\mu)}/\sqrt{r}) = -a + \frac{2C}{\sqrt{r}}\theta(l^{(\mu)}/\sqrt{r}) \quad (4-215)$$

$$\begin{aligned} F_2(l^{(\mu)}) &= \int_{-\infty}^{-l^{(\mu)}/\sqrt{r}} [-a(l^{(\mu)} + \sqrt{r}t) - C]^2 \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt + \\ &\quad \int_{-l^{(\mu)}/\sqrt{r}}^{\infty} [-a(l^{(\mu)} + \sqrt{r}t) + C]^2 \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt \end{aligned}$$

$$= a^2 [(l^{(\mu)})^2 + r] - 2ac \{ l^{(\mu)} [1 - 2\psi(l^{(\mu)}/\sqrt{r})] + 2\sqrt{r} \theta(l^{(\mu)}/\sqrt{r}) \} + C^2 \quad (4-216)$$

其中函数  $\psi(\cdot)$  用式 4-187 计算, 函数  $\theta(\cdot)$  用式 (4-179) 计算。当  $r$  足够小时  $\psi(l^{(\mu)}/\sqrt{r})$  以及  $\theta(l^{(\mu)}/\sqrt{r})$  与其他项相比可略之不计。为计算记忆容量 取  $l^{(\mu)} = 1$  即得到

$$F_1(l^{(\mu)}) = -a + C, \quad F_1(l^{(\mu)}) = -a, \quad F_2(l^{(\mu)}) = a^2(1+r) - 2aC + C^2 \quad (4-217)$$

将其代入式 (4-203) 即得到

$$L^{(\mu)} = 1 - a + C, \quad B = -ar, \quad \sigma^2 = r(1+C-2a)^2 + a^2r^2 \quad (4-218)$$

将这些值代入式 (4-189) 且因为  $l^{(\mu)} = 1$  所以  $d = 0$  即得到

$$\langle \tilde{d} \rangle = \psi\left(\frac{L^{(\mu)} + B}{\sigma}\right) = \psi(G), \quad G = \frac{1+C-a(1+r)}{[r(1+C-2a)^2 + a^2r^2]^{\frac{1}{2}}} \quad (4-219)$$

同前文所述相同 为使  $\langle \tilde{d} \rangle$  尽可能小 应使  $G$  尽可能大。现在设  $r$  取定值, 在此条件下求参数  $a$  和  $C$  使  $G$  达最大值。为此设  $u = 1 + C$  则有

$$G = \frac{u - a(1+r)}{[r(u-2a)^2 + a^2r^2]^{\frac{1}{2}}} \quad (4-220)$$

由  $dG/du = 0$  可求得使  $G$  值达最大的  $u$  值为

$$u = 1 + C = a \frac{2-r}{1-r} \quad (4-221)$$

当  $r$  足够小时 此式可近似为

$$1 + C = 2a \quad (4-222)$$

若  $C = 0$  则  $a = 0.5$  此结果与前文一致。将式 (4-222) 代入式 (4-220) 即得  $C$  和  $a$  满足最佳关系时的  $G$  值为

$$G = \frac{1-r}{r} \quad (4-223)$$

当  $r$  足够小时  $G \approx r^{-1}$ 。这样立即可确定相对记忆容量  $R$  为保证  $(G^2/2) \geq \ln N$  成立的  $r$  上限值, 由此即得到  $R$  和  $M_0$  值为

$$R = \frac{1}{\sqrt{2\ln N}}, \quad M_0 = NR = \frac{N}{\sqrt{2\ln N}} \quad (4-224)$$

当修正 HNN 的  $\varphi(\cdot)$  由式 (4-173) 确定且其参数  $a$  和  $C$  满足式 (4-222) 时 ( $a = 0.5, C = 0$  是其中一个特例), 其记忆容量即由式 (4-223) 确定。当  $a$  和  $C$  取满足式 (4-222) 的不同值时, 记忆容量相同而吸引域不同, 下面讨论此项内容。

## 2. 吸引域的计算

如前文所述, 一个 HNN 的吸引域用其吸引半径  $D$  描述。如用一个存储项  $\mathbf{X}^{(\mu)}$  的缺损样本  $\mathbf{X}^{(\mu)}$  作为  $\mathbf{X}(0)$  来一步提取  $\mathbf{X}^{(\mu)}$  若  $\mathbf{X}^{(\mu)}$  与  $\mathbf{X}^{(\mu)}$  之间的距离  $d \leq D$  时 即可实现正确提取。所谓正确是指提取出的  $\tilde{\mathbf{X}} = \mathbf{X}(1)$  与  $\mathbf{X}^{(\mu)}$  之间的距离  $\tilde{d}$  满足  $\langle \tilde{d} \rangle \leq 1/N$ 。  $D$  既取决于 HNN 的结构和参数, 又取决于装载率  $r$ 。对于本节讨论的内容, HNN 都可用  $a$  和  $C$  两个参数描述 当  $a = C = 0$  即为标准 HNN 若  $a$  和 / 或  $C$  不为 0 则为修正 HNN。一俟这些参数给定后,  $D$  即是  $r$  的函数。根据记忆容量的定义 当  $r = R$  时  $D = 0$  这时每个存储项



$\mathbf{X}^{(\mu)}$  的吸引域就是其本身。当  $r < R$  时,  $D$  随着  $r$  的减小而增大, 这时各  $\mathbf{X}^{(\mu)}$  的吸引域是为其中心且半径为  $D$  的超球。当  $r \rightarrow 0$  时,  $D$  增加到其最大值  $1/2$ 。虽然这种规律对于不同的参数  $a$  和  $C$  是相同的, 但是  $D$  随  $r$  的减小而增大的速度是大不相同的。这时需要计算并选择最佳的  $a$  和  $C$  参数, 使得在一定的  $r$  值时  $D$  达到尽可能大的值, 或在一定的  $D$  值下使  $r$  达到最大, 二者等效。

推导计算仍基于式 (4-189)。与记忆容量计算不同的是, 在计算吸引域时,  $\mathbf{X}^{(\mu)} = \mathbf{X}(0)$  与  $\mathbf{X}^{(\mu)}$  之间的距离  $d$  不再等于 0 相应地交迭  $l^{(\mu)} = 1 - 2d$  不再等于 1。借助于式 (4-203) 和式 (4-214) ~ 式 (4-216) 当  $r$  足够小时, 忽略掉  $\psi(l^{(\mu)}/\sqrt{r})$  和  $\theta(l^{(\mu)}/\sqrt{r})$ , 则式 (4-189) 中的  $L^{(\mu)} + B$  和  $L^{(\mu)} - B$  可用下式计算:

$$L^{(\mu)} \pm B = (1 - 2d)(1 - a) + C \mp ar \quad (4-225)$$

如  $C$  和  $a$  满足最大记忆容量关系式 (4-222) 即  $C = 2a - 1$  则此式成为

$$L^{(\mu)} \pm B = 2d(a - 1) + a \mp ar \quad (4-226)$$

同样, 由式 (4-203) 和式 (4-214) ~ 式 (4-216) 可以求得

$$\sigma^2 = r[1 + C^2 + 3a^2(l^{(\mu)})^2 - 4acl^{(\mu)} + a^2 + 2Cl^{(\mu)} - 2a(l^{(\mu)})^2 - 2a + a^2r]$$

将  $l^{(\mu)} = (1 - 2d)$  以及  $C = 2a - 1$  代入上式, 即可得

$$\sigma^2 = r[a^2r + 4(1 - a)^2d - 4a(2 - 3a)d^2] \quad (4-227)$$

注意式 (4-226) 中  $a \mp ar = a(1 \mp r)$  当  $r$  充分小时  $(1 \mp r) \approx 1$  这时该式可近似为

$$L^{(\mu)} \pm B \approx a - 2(1 - a)d \quad (4-228)$$

将式 (4-227) 和 (4-228) 代入式 (4-189) 并且用式 (4-189) 的近似式来计算  $\psi(\cdot)$  则得到

$$\langle \tilde{d} \rangle \approx \psi(G) \approx \frac{\theta(G)}{G}, \quad G = \frac{a - 2(1 - a)d}{\{r[a^2r + 4(1 - a)^2d - 4a(2 - 3a)d^2]\}^{\frac{1}{2}}} \quad (4-229)$$

利用此式可以描绘出对于不同的参数  $a$  取值 (相应的  $C$  值由式 (4-222) 确定) 当装载率  $r$  取各种值时  $\langle \tilde{d} \rangle \sim d$  变化曲线。图 4-10 所示是  $r = 0.2$  时,  $a$  和  $C$  取三组不同值的  $\langle \tilde{d} \rangle \sim d$  曲线。当  $r$  取 0.2 左右其他值时, 也可以得到大致相似的曲线。这些曲线的特点是: 当  $d = 0.5, \langle \tilde{d} \rangle = 0.5; \langle \tilde{d} \rangle$  随  $d$  的下降而单调下降; 当  $d = 0, \langle \tilde{d} \rangle = 0$ 。当  $r$  值一定时,  $\langle \tilde{d} \rangle$  随  $d$  的下降速度随参数  $a$  及相应  $C$  的改变而变化。当  $a$  由取负值变为取正值并增加到  $a = 1$  时, 下降速度单调增加; 当  $a$  由 1 再增加时下降速度缓慢减小。即  $a = 1$  时下降速度最快。从使  $\langle \tilde{d} \rangle$  最小的角度看, 应取的最佳参数组是  $C = 1, a = 1$ 。当  $C, a$  一定时,  $\langle \tilde{d} \rangle$  随  $d$  的下降速度随  $r$  的减小而增加。这样对于不同的装载率  $r, \langle \tilde{d} \rangle$  下降至  $(1/N)$  的  $d$  值不同。 $r$  值越小, 此  $d$  值越大。

这样, 就可以讨论一个 HNN 的装载率与吸引半径之间的关系。对于任何  $[0, 1/\sqrt{2\ln N}]$  范围内的  $r$  取值  $\hat{r}$  其吸引半径  $D$  可由下式确定:

$$\langle \tilde{d} \rangle \big|_{r=\hat{r}, d=D} = \frac{1}{N} \quad (4-230)$$

将式 (4-229) 代入此式并且在两侧取对数, 即得

$$\ln \theta(G) - \ln G + \ln N = 0, \quad G = G \big|_{r=\hat{r}, d=D}$$

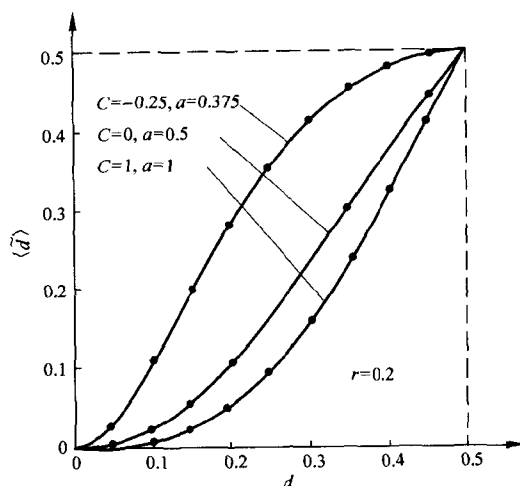


图 4-10  $\langle \tilde{d} \rangle \sim d$  曲线

如用式(4-179)计算  $\theta(G)$  并且在  $N$  足够大时使此式成立的  $G$  足够大, 这时可略去

$\frac{1}{2} \ln 2\pi - \ln G$  项 即得

$$G^2 |_{r=\hat{R}, d=D} = 2 \ln N \quad (4-231)$$

若取最佳参数  $a = 1$  则由式(4-229) 得到

$$G^2 |_{r=\hat{R}, d=D} = \frac{1}{\hat{R}(\hat{R} + 4D^2)} = 2 \ln N \quad (4-232)$$

此式可写成下列关于  $\hat{R}$  的二次方程式:

$$\hat{R}^2 + 4D^2 \hat{R} - \frac{1}{2 \ln N} = 0$$

由此式可求得  $\hat{R}$  的两个解, 一正一负, 可用的是正值解

$$\hat{R} = 2 \left\{ \left[ D^4 + \frac{1}{4} \frac{1}{2 \ln N} \right]^{\frac{1}{2}} - D^2 \right\} \quad (4-233)$$

设  $\hat{R} = \rho / \sqrt{2 \ln N}$ ,  $0 \leq \rho \leq 1$  则可求得

$$D = \frac{1}{2} \left( \frac{1 - \rho^2}{\rho} \right)^{\frac{1}{2}} \left( \frac{1}{2 \ln N} \right)^{\frac{1}{4}} \quad (4-234)$$

可以看到 当  $\hat{R} = R = 1/\sqrt{2 \ln N}$ ,  $\rho = 1$  则  $D = 0$ . 当  $\hat{R} < R$  时 相应的  $D$  随  $\rho$  的减小而增加. 当  $\hat{R} = 1/2 \ln N$  即  $\rho = 1/\sqrt{2 \ln N}$  时,  $D$  值为

$$D = \frac{1}{2} \left[ 1 - \frac{1}{2 \ln N} \right]^{\frac{1}{2}}$$

当  $N$  较大时,  $D \approx 0.5$ . 当  $\rho < 1/\sqrt{2 \ln N}$  时吸引半径不可能大于 0.5 即保持为  $0.5 - \epsilon$ .

$0 < \epsilon \ll 1$ . 这样 可以描绘出一条修正 HNN 取参数  $a = C = 1$  时  $D$  随  $\hat{R}$  变化的曲线. 图 4-11 所示为  $N = 1000$  时  $D \sim \hat{R}$  曲线的示例. 可以看到 取最佳参数值的修正 HNN 较标准 HNN 无论在记忆容量还是吸引域的扩大方面都有十分可观的改善, 随着  $N$  值的加大,

改善程度也继续加大。这里给出的是步运行的结果。4.5.1 小节已给出，在二步运行或非同步多步运行的情况下，对于标准 HNN 只要满足  $0 \leq R < 1/(2\ln N)$  则吸引半径  $D$  恒为  $0.5 - \epsilon, 0 < \epsilon \ll 1$ 。对于修正 HNN 二步(同步)运行和多步(非同步)运行情况下  $D \sim R$  关系为当  $0 \leq R < 1/(2\ln N)$  时仍有  $D = 0.5 - \epsilon, 0 < \epsilon \ll 1$  但是当  $1/(2\ln N) \leq R < 1/\sqrt{2\ln N}$  时  $D$  如何随  $R$  而变化尚待研究。

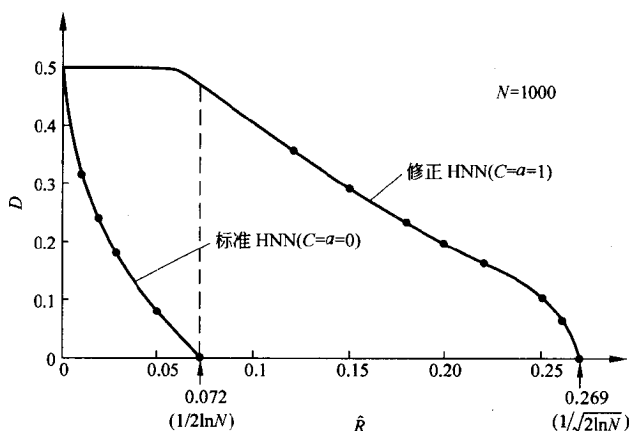


图 4-11  $N = 1000$  时  $D \sim R$  曲线

以上两小节讨论的都是基于各记忆项独立这一假设下，HNN 的权矩阵  $\mathbf{W}$  由 Hebb 学习律来求得的。而在大量实际问题中，这一假设并不成立，这时采用简单的 Hebb 学习律来求  $\mathbf{W}$  其效果并不好。以下两节将讨论两种修正学习算法。

### 4.5.3 HNN 学习算法的改进方案

设有  $M$  个记忆项  $\mathbf{X}^{(m)} = [x_1^{(m)}, x_2^{(m)}, \dots, x_N^{(m)}]^T, m = 1 \sim M$ 。各记忆项之间以及每个记忆项内各分量之间并非独立。对各分量概率分布也不做任何假设。现在要求构成一个含  $N$  个神经元的 HNN 来存储这  $M$  个记忆项，规定其权矩阵  $\mathbf{W}$  为 0 主对角线对称阵（见式 (4-160)）且规定按非同步方式运行（见式 (4-161)）。现在的问题是如何确定  $\mathbf{W}$  以使得每个  $\mathbf{X}^{(m)}$  都是吸引子而且其周围的吸引域尽可能大。此时采用简单 Hebb 学习律来求得的  $\mathbf{W}$ （见式 4-164）往往效果不佳。下面先给出在满足这两项主要要求而构成学习算法时用到的两个基本定理。

**定理 1**  $\mathbf{X}^{(m)}, m = 1 \sim M$  都是吸引子的充分条件是

$$F_i^{(m)} = \left( \sum_{j=1}^N w_{ij} x_j^{(m)} \right) x_i^{(m)} > 0, \quad \forall i = 1 \sim N, m = 1 \sim M \quad (4-235)$$

**证明** HNN 中任一状态  $\mathbf{X}$  成为吸引子的条件由式 4-163 给出。式 (4-235) 成立时式 (4-163) 成立，所以各存储项  $\mathbf{X}^{(m)}$  皆为吸引子。证完。文献 [72]、[73]、[74] 均述及此定理。

**定理 2** 对于  $\mathbf{X}^{(m)}$  的缺损样本  $\hat{\mathbf{X}}^{(m)}$  若下列条件满足

$$F_i^{(m)} \geq F > 0, \quad i = 1 \sim N, \quad m = 1 \sim M \quad (4-236)$$

$$d_H(\mathbf{X}^{(a)}, \mathbf{X}^{(b)}) \leq D = F/(2W_{\max}) \quad (4-237)$$

其中  $F$  是某个正数。 $d_H(\mathbf{X}^{(a)}, \mathbf{X}^{(b)}) = Nd(\mathbf{X}^{(a)}, \mathbf{X}^{(b)})$  是  $\mathbf{X}^{(a)}, \mathbf{X}^{(b)}$  之间的汉明距离,  $d(\mathbf{X}^{(a)}, \mathbf{X}^{(b)})$  是相对汉明距离 (见式 4-167))。 $W_{\max}$  是诸  $w_{ij}$  绝对值中最大者, 即

$$W_{\max} = \max_{i,j} |w_{ij}| \quad (4-238)$$

则当设置 HNN 初始状态为  $\mathbf{X}(0) = \hat{\mathbf{X}}^{(m)}$  且按照非同步方式 (式 4-161)) 运行网络,  $N$  个神经元依次调节一次后, 即得到  $\mathbf{X}(N) = \mathbf{X}^{(m)}$ 。即可由  $\mathbf{X}^{(m)}$  实现对  $\mathbf{X}^{(m)}$  的正确提取。

此定理的证明可以从文献 [74] 中找到。定理 1 和定理 2 表明, 不管各  $\mathbf{X}^{(m)}$  之间或之内是否独立, 只要式 4-236) 给出的条件得到满足, 则各  $\mathbf{X}^{(m)}$  都是吸引子且吸引半径  $D$  由式 (4-237) 确定。现在需构成一种求  $\mathbf{W}$  的学习算法, 所求得的  $\mathbf{W}$  应使  $F$  尽可能大且  $W_{\max}$  尽可能小, 从而使得  $D$  尽可能大。现在常取的方案是在保持  $W_{\max}$  一定的条件下, 求  $\mathbf{W}$  使  $F$  尽可能大。下面介绍其中的几种。

### 1. 局部迭代学习算法 [75]

算法框架如下。

(1) 设置  $F$  值, 设置最大迭代次数  $K_{\max}$ 。

(2) 设置各权初值  $w_{ij}(0)$  为非 0 随机值,  $i \neq j$  为 0 值,  $i = j$ 。

(3) 按节拍  $k = 0, 1, 2, \dots$  从  $\mathbf{X}^{(m)}, m = 1 \sim M$  中依次选出一个样本作为进行学习的样本, 记为  $\mathbf{X}(k)$  (当  $M$  个样本全选过后再从头选起)。对各  $w_{ij}$  进行迭代调整计算:

$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}(k), i, j = 1 \sim N$ 。 $\Delta w_{ij}(k)$  用下式计算:

$$\Delta w_{ij}(k) = \begin{cases} [\epsilon_i(k) + \epsilon_j(k)]x_i(k)x_j(k)/\sqrt{N}, & i \neq j \\ 0, & i = j \end{cases} \quad (4-239)$$

$x_i(k), x_j(k)$  是  $\mathbf{X}(k)$  第  $i, j$  分量。 $\epsilon_i(k), \epsilon_j(k)$  称为掩盖系数, 用下式计算:

$$\epsilon_l(k) = \theta \left[ F \cdot \left( \sum_{p=1}^N w_{lp}^2(k) \right)^{\frac{1}{2}} - F_l(k) \right], \quad F_l(k) = \left( \sum_{p=1}^N w_{lp}(k)x_p(k) \right)x_l(k), \quad l = i, j \quad (4-240)$$

其中

$$\theta[\xi] = \begin{cases} 1, & \xi > 0 \\ 0, & \xi \leq 0 \end{cases} \quad (4-241)$$

(4) 检验  $\epsilon_i(k) = 0, \forall i = 1 \sim N$  是否成立。若成立, 则算法收敛, 以  $w_{ij}(k)$  作为计算结果。若不成立, 则检验  $k < K_{\max}$  是否成立。若成立, 则继续进行迭代。若不成立, 则算法不收敛。

(5) 若算法收敛, 则增大  $F$  值并进行新一轮迭代计算, 直至  $F$  增大到使算法不收敛为止, 从而求得最大  $F$  值。若算法不收敛, 则减小  $F$  值并进行新一轮迭代计算。若仍不收敛, 再降低  $F$  值。若  $F$  降至 0 仍不收敛, 则问题无解。

文献 [75] 严格证明了, 只要存在权矩阵  $\mathbf{W} = (\hat{w}_{ij})$  使得下式成立:

$$F_i^{(m)} = \left( \sum_{j=1}^N \hat{w}_{ij}x_j^{(m)} \right)x_i^{(m)} > (F + \delta) \left( \sum_{j=1}^N \hat{w}_{ij}^2 \right)^{\frac{1}{2}}, \quad i = 1 \sim N, m = 1 \sim M \quad (4-242)$$

其中  $0 < \delta \ll 1$ 。那么,只要  $K_{\max}$  是充分大的有限值,该算法一定能使各  $w_{ij}$  收敛到  $w_{ij}$ 。注意,若  $\hat{W} = (\hat{w}_{ij})$  满足式(4-242)则  $\alpha \hat{W} = (\alpha \hat{w}_{ij})$  也必满足此式,即解的尺度伸缩不改变其性质。特别是当下列条件满足时:

$$\left( \sum_{j=1}^N \hat{w}_{ij}^2 \right)^{\frac{1}{2}} = 1 \quad (4-243)$$

式(4-242)成为

$$F_i^{(m)} > F > 0, \quad i = 1 \sim N, m = 1 \sim M$$

此式与(4-237)一致。而由式(4-243)可知,各  $|\hat{w}_{ij}|$  的最大值为  $W_{\max} \leq 1$ 。这样,由式(4-239)可推断,由这个  $\hat{W}$  所确定的吸引半径为  $D \geq F/(2W_{\max}) = F/2$ 。当各  $|w_{ij}|$  都增大至  $\alpha$  倍时,式(4-242)成为

$$F_i^{(m)} > \alpha F > 0, \quad i = 1 \sim N, m = 1 \sim M$$

而各  $|w_{ij}|$  最大值为  $W_{\max} \leq \alpha$ , 这时吸引半径仍满足  $D \geq F/2$ 。这个算法的核心是引进了

掩盖系数  $\epsilon_i(k)$  和  $\epsilon_j(k)$ 。由式(4-240)可见,若  $\left( \sum_{p=1}^N w_{ip}^2(k) \right)^{\frac{1}{2}} = 1$ , 那么只有当  $F > F_i(k)$  时  $\epsilon_i(k) = 1$ ; 反之  $\epsilon_i(k) = 0$ 。也就是说,当  $F_i(k) > F$  的条件不满足时才对  $w_{ij}$  进行调整;一旦满足就不再调整。从而使式(4-236)得到实现。此外,这个算法对于  $W_{\max}$  没有予以直接限制,而是通过式(4-243)来予以间接限制的。由于  $F$  不能事先确定,只能在迭代计算中通过算法是否收敛来确认。

文献[74]对此算法作了一种修正。在上述框架的步骤中不是在各  $X^{(m)}$  中依次轮流选择一个出来作为  $X(k)$ ,而是首先计算各个  $F_i^{(m)}(k) = \left( \sum_{j=1}^N w_{ij}(k) x_j^{(m)} \right) x_i^{(m)}$ 。若

$$F_i^{(\mu)}(k) = \min_{m=1 \sim M, i=1 \sim N} [F_i^{(m)}(k)]$$

则以  $X^{(\mu)}$  作为  $X(k)$  并仍按式(4-239)对权进行调整。这称为最小覆盖学习规则。

文献[76]指明,采用这种局部迭代学习算法的 HNN 其绝对记忆容量  $M_0$  最高可达至  $2N$ 。

## 2. 简化局部迭代学习算法和记忆项的动态提取<sup>[77]</sup>

此算法不用式(4-243)来间接限制  $W_{\max}$  而直接设定  $W_{\max}$ 。其算法框架如下。

(1) 设置  $F$  值,设置  $K_{\max}$ , 设置  $W_{\max}$ 。

(2) 设置各权初值为  $w_{ij}(0) = 0, \forall i, j = 1 \sim N$ 。

(3) 按节拍  $k = 0, 1, 2, \dots$  依次从  $X^{(m)}, m = 1 \sim M$  中依次选出一个样本作为  $X(k)$ 。进行迭代计算

$$w_{ij}(k+1) = G[w_{ij}(k) + \Delta w_{ij}(k)], \quad i, j = 1 \sim N \quad (4-244)$$

其中

$$G[u] = \begin{cases} -W_{\max}, & u < -W_{\max} \\ u, & -W_{\max} \leq u \leq W_{\max} \\ W_{\max}, & u > W_{\max} \end{cases} \quad (4-245)$$

$\Delta w_{ij}(k)$  仍用式(4-239)计算,只是其中的掩盖系数改用下式计算:

$$\varepsilon_i(k) = \theta[F - F_l(k)], \quad F_l(k) = \left( \sum_{p=1}^N w_{lp} x_p(k) \right) x_l(k), \quad l = i, j \quad (4-246)$$

其中  $\theta[\cdot]$  仍用式 (4-241) 计算。

(4) 若  $\varepsilon_i(k) = 0, i = 1 \sim N$  则算法收敛, 令  $F$  值增大为  $F + \Delta F, \Delta F > 0$  转至 (3) 进行新一轮迭代计算。若  $\varepsilon_i(k)$  不皆为 0 且  $k = K_{\max}$  则算法不收敛。 $F$  值减小为  $F - \Delta F$ , 转至 (3) 再算。

这一算法的收敛性尚无严格证明, 但在实际应用中可行。此外, 文献 [77] 还提出一种动态提取算法来扩大吸引半径  $D$ 。按照本节所述的各种局部迭代学习算法, 所求得的权矩阵  $\mathbf{W}$  使得吸引半径  $D = F/(2W_{\max})$  (见式 4-237)。如果用式 (4-162) 作为网络中每一状态  $\mathbf{X}$  的能量函数  $E(\mathbf{X})$  的定义, 那么文献 [77] 证明了, 在每一个以  $\mathbf{X}^{(m)}, m = 1 \sim M$  为中心, 半径为  $ND$  的超球中  $E(\mathbf{X})$  只有一个极小点, 即  $\mathbf{X}^{(m)}$ 。所以当  $\mathbf{X}(0) = \hat{\mathbf{X}}^{(m)}$  在以  $\mathbf{X}^{(m)}$  为中心且半径为  $NF/(2W_{\max})$  的超球中时, 一定可以通过非同步运行 (式 (4-161)) 收敛到  $\mathbf{X}^{(m)}$ 。文献 [77] 还证明, 在每一个以  $\mathbf{X}^{(m)}, m = 1 \sim M$  为中心, 半径为  $NF/W_{\max}$  的超球中,  $E(\mathbf{X})$  除了  $\mathbf{X}^{(m)}$  是其全局最小点外, 还可能存在若干相应于虚假吸引子的局部极小点。它们与存储项无关且这些点上的  $E(\mathbf{X})$  值高于相应中心  $\mathbf{X}^{(m)}$  上的  $E(\mathbf{X})$  值。若在运行时刻  $t$ , 式 (4-161) 中选中进行调整的神经元编号是  $i = l$ , 那么由式 (4-162) 可算出调整前后能量函数的变化量  $\Delta E(t)$  为

$$\Delta E(t) = E(t+1) - E(t) = -2 \left| \sum_{j=1}^N w_{lj} x_j(t) \right| \quad (4-247)$$

当  $l$  不同时, 能量函数的降低量不同。如果  $\mathbf{X}(0) = \hat{\mathbf{X}}^{(m)}$  在某个局部极小点附近, 而  $t = 0$  时刻所选的  $l$  相应的  $|\Delta E(t)|$  较小, 这时网络状态就会趋近该局部极小点并最终收敛于相应的虚假吸引子。相反, 如果所选的  $l$  相应的  $|\Delta E(t)|$  很大, 就有可能在运行过程中跨越过局部极小点而收敛于全局最小点  $\mathbf{X}^{(m)}$ 。所取的策略是设置一个随运行节拍  $t$  而改变的阈值  $T(t)$ , 当  $t$  较小时  $T(t)$  较大, 随着  $t$  的增大  $T(t)$  逐渐减小。在每个时刻  $t$  所选的被动调神经元  $l(t)$  不是随机或依次, 而是选相应  $|\Delta E(t)| > T(t)$  者。模拟实验及针对字符识别所作的研究证实了这种算法可以将吸引半径从式 (4-237) 给出的  $D = F/(2W_{\max})$  扩展为  $D = F/W_{\max}$ 。

#### 4.5.4 HNN 学习算法的另一种改进方案

文献 [78] - [80] 给出另一种 HNN 的迭代式权学习算法。算法框架如下。

(1) 置初值  $w_{ij}(0) = 0, \forall i, j = 1 \sim N$ 。

(2) 按节拍  $k = 0, 1, \dots, M-1$  依次从  $\mathbf{X}^{(m)}, m = 1 \sim M$  中选出一个样本作为  $\mathbf{X}(k)$  来进行权学习。令  $w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}(k)$ ,

$$\Delta w_{ij}(k) = \begin{cases} \frac{1}{N} [x_i(k)x_j(k) - x_i(k)h_{ji}(k) - x_j(k)h_{ij}(k)], & i \neq j \\ 0, & i = j \end{cases} \quad (4-248)$$

其中

$$h_{ji}(k) = \sum_{\substack{l=1 \\ l \neq i, j}}^N w_{jl}(k-1)x_l(k), \quad h_{ij}(k) = \sum_{\substack{l=1 \\ l \neq i, j}}^N w_{il}(k-1)x_l(k) \quad (4-249)$$

(3) 以  $w_{ij}(M)$ ,  $\forall i, j = 1 \sim N$  作为学习结果输出。

此学习算法有以下特点。

(1) 若各个记忆样本独立 则 HNN 的相对记忆容量为  $R = 1/\sqrt{2\ln N}$ <sup>[78]</sup> 这和 4.5.2 小节中通过改变神经元函数所得到的结果一致。

(2) 与标准 HNN 相比, 不仅记忆容量扩大很多而且吸引域的大小、形状和分布均匀度都有很大改善。吸引域的形状是指每个存储项周围各个方向上的吸引距离有所不同, 当吸引距离各向一致时是一个超球, 不一致时是一个超椭球, 可以用非对称度 (skew) 的大小来描述这种不一致性的程度。此前各节用吸引半径来描述吸引域是忽略了这种非对称度并且以吸引距离最小者作为吸引半径。分布均匀度是指各记忆项周围吸引域的大小差异, 若差异小则分布均匀, 反之则否。文献[79]针对一个  $N = 250$  的 HNN 做了一个有趣的模拟实验 在不同装载率  $r$  下 (各记忆项独立) 用标准 HNN 学习算法和本节的修正学习算法求  $\mathbf{w}$ 。然后求每个记忆项的吸引半径和非对称度, 从而得到下面一些结论。

① 标准 HNN 的相对记忆容量为  $R = 1/(2\ln N) = 0.09$ , 绝对记忆容量为  $M_0 = NR = 22$ 。修正算法则为  $R = 0.30, M_0 = 75$ 。

当装载率  $r = 0.22$  相应存储记忆项个数  $M = 55$ 。对于标准 HNN 所有存储项的吸引半径皆为 0 或为非吸引子。对于修正 HNN, 所有存储项都是吸引子, 它们的吸引半径值分布在 0.06 ~ 0.35 之间。当装载率  $r = 0.12$  相应  $M = 30$ 。标准 HNN 的 30 个存储项中有 5 项的吸引半径为 0 或非吸引子。修正 HNN 的 30 个存储项皆为吸引子, 其吸引半径值分布在 0.22 ~ 0.38 之间。当装载率  $r = 0.08$  相应  $M = 19$ 。标准 HNN 的 19 个存储项中有 1 个吸引半径为 0 其他项吸引半径在 0.16 ~ 0.34 之间。对于修正 HNN, 19 项的吸引半径都在 0.3 ~ 0.4 之间。可以看到  $r$  越大则修正 HNN 的优势越明显。

对于修正 HNN, 无论各记忆项的吸引半径取较大或较小值, 其吸引域的非对称度都不大。而对于标准 HNN, 当吸引半径较大时非对称度较小; 反之, 当吸引半径较小时非对称度很大。

(3) 当各记忆项之间存在相关性时, 修正算法优于标准算法。文献[80]针对一个  $N = 500$  的 HNN, 研究其存储具有时间相关性的记忆项时两种算法记忆容量的大小随相关性大小的变化。所存储的记忆项记为  $\mathbf{X}(k), k = 0, 1, 2, \dots$ 。对于任何  $i = 1 \sim N, x_i(0)$  取 -1 或 1 的概率都等于 1/2 而  $x_i(k)$  与  $x_i(k+1)$  相同的概率为  $\rho$  即

$$Pr[x_i(k) = x_i(k+1)] = \rho, \quad \frac{1}{2} \leq \rho \leq 1, \quad \forall i, k$$

当  $\rho = 1/2$  时相邻记忆项之间完全不相关, 当  $\rho = 1$  则二者完全一致。模拟实验取 30 种不同的  $\mathbf{X}(0)$  形成 30 组不同的记忆项序列, 用两种算法求相应的  $\mathbf{w}$  和绝对记忆容量  $M_0$ 。图 4-12 给出了这两种算法所求得的 30 个  $M_0$  的最大、最小和平均值以及分布区间随记忆项序列相邻项相关系数  $C$  的变化。 $C$  可由  $\rho$  求得如下:

$$C = \langle x_i(k)x_i(k+1) \rangle / [\langle x_i^2(k) \rangle \langle x_i^2(k+1) \rangle]^{1/2} = 2\rho - 1$$

图中曲线 ① 相应于修正算法 曲线 ② 相应于标准 HNN 算法。可以看到 当  $C$  达到 0.4 左

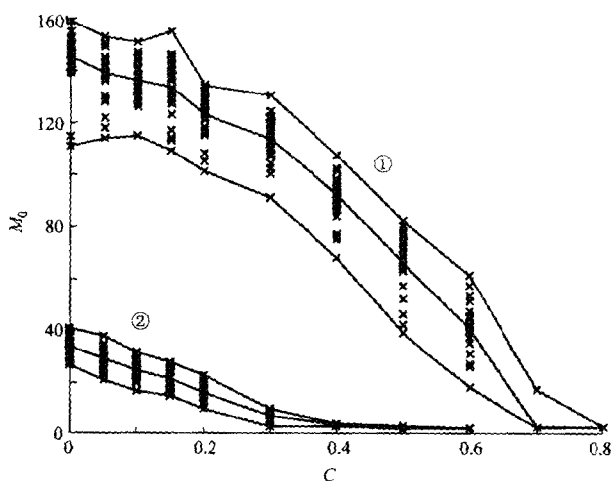


图 4-12  $C \sim M_0$  曲线

右时，标准算法的绝对记忆容量已接近于 0 而修正算法的  $M_0$  仍可达 70 ~ 100。

(4) 修正算法所形成的  $W$  与各记忆项呈现给 HNN 的次序有关。此外 如果给予新的记忆项 那么 HNN 通过对其学习而“遗忘”掉早期学习的旧记忆项。这些都是有待研究的问题。

上面讨论的几种 HNN 修正学习算法都采取迭代计算，因此算法简单且计算量很小。现在还有一种根据“似逆规则”(pseudo-inverse rule) 构成的求权矩阵  $W$  的算法 它既不依赖于 Hebb 学习律又不取迭代计算<sup>[86]</sup> 其  $M_0$  值等于 HNN 的神经元数  $N$ 。其问题是计算量巨大 特别是当  $N$  很大时，需完成  $N$  维方阵求逆 计算量过大，另外 当  $M_0$  接近  $N$  时，吸引域迅速趋近于 0。

关于 HNN 记忆容量和吸引域的研究还可以参考文献[81] ~ [85]。

## 4.6 双向联想记忆及其他联想记忆神经网络

### 4.6.1 BAM

双向联想记忆 (BAM) 又称为异联想记忆 (hetero-AM)，以区别于前节的自联想记忆 (auto-AM) 它是 B. Kosko 在 20 世纪 80 年代后期提出的<sup>[87,88]</sup>。其思路是用一个神经网络来存储由  $M$  对向量  $\{X^{(m)} = [x_1^{(m)}, \dots, x_N^{(m)}]^T, Y^{(m)} = [y_1^{(m)}, \dots, y_P^{(m)}]^T\}, m = 1 \sim M$  构成的  $M$  个记忆项。其中  $x_i^{(m)}, i = 1 \sim N$  和  $y_j^{(m)}, j = 1 \sim P$  只能取值 1 或 -1。网络由两层构成，一层含  $N$  个神经元 其输出记为  $x_i, x_i \in \{1, -1\}$ ；另一层含  $P$  个神经元 其输出记为  $y_j, y_j \in \{1, -1\}$ 。每层各神经元的输出只馈送到另一层各神经元的输入而不馈送至本层的各个神经元。此网络结构如图 4-13 所示。网络按照离散时间节拍  $k = 0, 1, 2, \dots$  运行，网络中各神经元的输出作为  $k$  的函数 记为  $x_i(k), y_j(k)$ 。任时刻  $k$  可用  $\{X(k), Y(k)\}$



描述网络所处的状态。网络运行方式分成两类。第一类，给定初值  $x_i(0), i = 1 \sim N$  当  $k = 0, 2, 4, \dots, x_i(k)$  不变而改变  $y_j(k)$ ；当  $k = 1, 3, 5, \dots$ ，

则  $y_j(k)$  不变而改变  $x_i(k)$ 。状态调整公式为

当  $k = 0, 2, 4, \dots$  时，

$$x_i(k+1) = x_i(k), \quad i = 1 \sim N;$$

$$y_j(k+1) = \operatorname{sgn}\left[\sum_{i=1}^N w_{ij}x_i(k)\right], \quad j = 1 \sim P \quad (4-250)$$

当  $k = 1, 3, 5, \dots$  时，

$$y_j(k+1) = y_j(k), \quad j = 1 \sim P;$$

$$x_i(k+1) = \operatorname{sgn}\left[\sum_{j=1}^P w_{ij}y_j(k)\right], \quad i = 1 \sim N \quad (4-251)$$

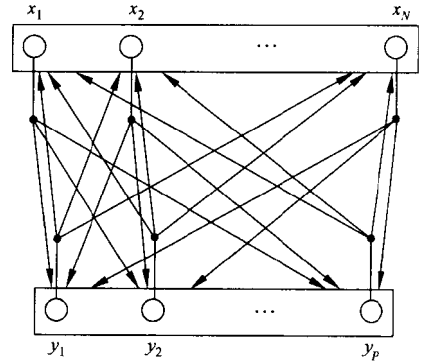


图 4-13 BAM 神经网络结构

其中  $\operatorname{sgn}[\cdot]$  按式 (4-159) 计算。第二类 给定初值  $y_j(0), j = 1 \sim P$  当  $k = 0, 2, 4, \dots, y_j(k)$  不变而调整  $x_i(k)$  当  $k = 1, 3, 5, \dots, x_i(k)$  不变而调整  $y_j(k)$ 。调整公式与第一类相同。上式中各个权  $w_{ij}$  构成一个  $N \times P$  维矩阵  $\mathbf{W} = (w_{ij})$ 。设  $\mathbf{X}(k) = [x_1(k), \dots, x_N(k)]^T$ ,  $\mathbf{Y}(k) = [y_1(k), \dots, y_P(k)]^T$  则式 (4-250) 和 (4-251) 可表示为

$$\mathbf{X}(k+1) = \mathbf{X}(k), \quad \mathbf{Y}(k+1) = \operatorname{sgn}[\mathbf{W}^T \mathbf{X}(k)], \quad k = 0, 2, 4, \dots \quad (4-252)$$

$$\mathbf{Y}(k+1) = \mathbf{Y}(k), \quad \mathbf{X}(k+1) = \operatorname{sgn}[\mathbf{W} \mathbf{Y}(k)], \quad k = 1, 3, 5, \dots \quad (4-253)$$

其中  $\operatorname{sgn}[\cdot]$  的定义见式 (4-170)。

网络状态的另一种调整公式是将式 (4-250) 和式 (4-251) 中的  $\operatorname{sgn}[\cdot]$  运算修正如下：

$$y_j(k+1) = \begin{cases} \operatorname{sgn}[\rho_j(k)], & \rho_j(k) = \sum_{i=1}^N w_{ij}x_i(k) \neq 0 \\ y_j(k), & \rho_j(k) = 0, \quad j = 1 \sim P \end{cases} \quad (4-254)$$

$$x_i(k+1) = \begin{cases} \operatorname{sgn}[\zeta_i(k)], & \zeta_i(k) = \sum_{j=1}^P w_{ij}y_j(k) \neq 0 \\ x_i(k), & \zeta_i(k) = 0, \quad i = 1 \sim N \end{cases} \quad (4-255)$$

这两种状态调整方法只有微小差异，在讨论一种网络的权学习算法前，需预先规定采取这两种方法中的哪一种。对于任一时刻  $k$  网络所处的状态  $\{\mathbf{X}, \mathbf{Y}\}$  为简化起见，省略了时间变量  $k$ ，可以定义相应的能量函数为

$$E(\mathbf{X}, \mathbf{Y}) = -\mathbf{X}^T \mathbf{W} \mathbf{Y} = -\mathbf{Y}^T \mathbf{W}^T \mathbf{X} \quad (4-256)$$

若  $k$  时刻的网络能量为  $E(k) = E(\mathbf{X}(k), \mathbf{Y}(k))$ ,  $k+1$  时刻为  $E(k+1) = E(\mathbf{X}(k+1), \mathbf{Y}(k+1))$ ，则无论采取这两种调整方法中的哪一种，都易于证明

$$E(k+1) \leq E(k), \quad \forall k \geq 0 \quad (4-257)$$

以第二种方法的式 (4-255) 为例，由式 (4-256) 可得

$$E(k+1) - E(k) = -\Delta \mathbf{X}^T(k) \mathbf{W} \mathbf{Y}(k) = -\sum_{i=1}^N \Delta x_i(k) \left[ \sum_{j=1}^P w_{ij} y_j(k) \right]$$

其中  $x_i(k) = x_i(k+1) - x_i(k)$  且  $x_i(k)$  只可能取  $+2, -2$  和  $0$  三种值。由式 (4-255), 若  $x_i(k) = 2$  则必有  $\zeta_i(k) = \sum_{j=1}^P w_{ij} y_j(k) > 0$  若  $\Delta x_i(k) = -2$  则必有  $\zeta_i(k) < 0$  因此恒有  $x_i(k) \zeta_i(k) \geq 0$  即证明式 (4-257) 成立。对于式 (4-254)、式 (4-250) 和式 (4-251) 也可作类似的证明。

无论取何初值 第一类 取  $\mathbf{X}(0)$  而  $\mathbf{Y}(0)$  任意 第二类 取  $\mathbf{Y}(0)$  而  $\mathbf{X}(0)$  任意 无论按何种权调整公式运行, BAM 经过有限多步 (设为  $K$  步) 运行后必然收敛到状态  $\{\mathbf{X}(K), \mathbf{Y}(K)\}$  即有  $\mathbf{X}(K+k) = \mathbf{X}(K)$  且  $\mathbf{Y}(K+k) = \mathbf{Y}(K), \forall k \geq 0$ 。此状态即为 BAM 诸定点 (稳定平衡点) 中的一个。至于收敛到哪一个定点, 则取决于初值  $\mathbf{X}(0)$  或初值  $\mathbf{Y}(0)$ 。

现在讨论用 BAM 存储  $M$  个记忆项  $\{\mathbf{X}^{(m)}, \mathbf{Y}^{(m)}\}, m = 1 \sim M$  的问题。为实现此任务, 应达到以下两项要求。

(1) 每一个记忆项  $\{\mathbf{X}^{(m)}, \mathbf{Y}^{(m)}\}$  都是 BAM 的定点。即当  $\mathbf{X}(0) = \mathbf{X}^{(m)}$  则经过有限多步 (设  $K$  步) 运行后 收敛到  $\mathbf{X}(K) = \mathbf{X}^{(m)}, \mathbf{Y}(K) = \mathbf{Y}^{(m)}$ 。或者当  $\mathbf{Y}(0) = \mathbf{Y}^{(m)}$  则经过有限多步运行后也能收敛到  $\mathbf{X}^{(m)}, \mathbf{Y}^{(m)}$ 。这样, 可以由  $\mathbf{X}^{(m)}$  提取出  $\mathbf{Y}^{(m)}$  或者反之, 由  $\mathbf{Y}^{(m)}$  提取出  $\mathbf{X}^{(m)}$ 。

(2) 在  $\mathbf{X}^{(m)}$  和  $\mathbf{Y}^{(m)}$  周围有尽量大的吸引域。若  $\hat{\mathbf{X}}^{(m)}$  或  $\hat{\mathbf{Y}}^{(m)}$  是吸引域内  $\mathbf{X}^{(m)}$  或  $\mathbf{Y}^{(m)}$  的缺损样样本 则当  $\mathbf{X}(0) = \hat{\mathbf{X}}^{(m)}$  或  $\mathbf{Y}(0) = \hat{\mathbf{Y}}^{(m)}$  时, 经过有限多步运行网络状态即收敛到  $\{\mathbf{X}^{(m)}, \mathbf{Y}^{(m)}\}$ 。这样, BAM 可同时实现记忆提取和去除噪声或污染。

和自联想记忆 HNN 的情况相似 用 HNN 实现 BAM 时 存储项的个数  $M$  吸引域的大小以及各存储项的性质是相互有关的。现在的问题是设计一种求权矩阵  $\mathbf{W}$  的学习算法, 针对于待存储各记忆项的性质使得吸引域大小一定时  $M$  达到最大值 或者  $M$  一定时吸引域达到最大。最早提出 BAM 的 B. Kosko 按照 Hebb 学习律采用外积和来求  $\mathbf{W}^{[87,88]}$  即对于  $M$  个记忆项  $\{\mathbf{X}^{(m)}, \mathbf{Y}^{(m)}\}, m = 1 \sim M, \mathbf{W}$  可用下式求得:

$$\mathbf{W} = \frac{1}{M} \sum_{m=1}^M \mathbf{X}^{(m)} [\mathbf{Y}^{(m)}]^T \quad (4-258)$$

文献[89] 证明了, 用此公式计算  $\mathbf{W}$  时, 各记忆项都是定点的充分条件可陈述如下:

已知  $\mathbf{X}^{(m)}$  与  $\mathbf{X}^{(\mu)}$  之间的汉明距离定义为

$$d_H(\mathbf{X}^{(m)}, \mathbf{X}^{(\mu)}) = \frac{1}{2} \sum_{i=1}^N |x_i^{(m)} - x_i^{(\mu)}| \quad (4-259)$$

那么  $\{\mathbf{X}^{(m)}, \mathbf{Y}^{(m)}\}$  成为定点的充分条件是

$$\min\{d_H(\mathbf{X}^{(m)}, \mathbf{X}^{(\mu)}), d_H(\mathbf{X}^{(m)}, -\mathbf{X}^{(\mu)})\} = He(\mathbf{X}^{(m)}, \mathbf{X}^{(\mu)}) \geq \frac{N}{2} \frac{(M-2)}{(M-1)}$$

以及

$$\min\{d_H(\mathbf{Y}^{(m)}, \mathbf{Y}^{(\mu)}), d_H(\mathbf{Y}^{(m)}, -\mathbf{Y}^{(\mu)})\} = He(\mathbf{Y}^{(m)}, \mathbf{Y}^{(\mu)}) \geq \frac{P}{2} \frac{(M-2)}{(M-1)},$$

$$\mu = 1 \sim M, \mu \neq m \quad (4-260)$$

而且这在大多数情况下也是一个必要条件。易于见到，如果  $M$  值较大 则  $(M-2)/(M-1) \approx 1$  这时式 (4-260) 可以用下式替代：

$$He(\mathbf{X}^{(m)}, \mathbf{X}^{(\mu)}) \geq \frac{N}{2}, \quad He(\mathbf{Y}^{(m)}, \mathbf{Y}^{(\mu)}) \geq \frac{P}{2}, \quad \mu, m = 1 \sim M, \quad \mu \neq m$$

还可以注意到  $He(\mathbf{X}^{(m)}, \mathbf{X}^{(\mu)})$  和  $He(\mathbf{Y}^{(m)}, \mathbf{Y}^{(\mu)})$  应满足

$$He(\mathbf{X}^{(m)}, \mathbf{X}^{(\mu)}) \leq \frac{N}{2}, \quad He(\mathbf{Y}^{(m)}, \mathbf{Y}^{(\mu)}) \leq \frac{P}{2}, \quad \mu, m = 1 \sim M, \quad \mu \neq m$$

为了使上列诸式都满足，下列条件应成立：

$$He(\mathbf{X}^{(m)}, \mathbf{X}^{(\mu)}) = \frac{N}{2}, \quad He(\mathbf{Y}^{(m)}, \mathbf{Y}^{(\mu)}) = \frac{P}{2}, \quad \mu, m = 1 \sim M, \quad \mu \neq m \quad (4-261)$$

在大多数情况下，这就是  $M$  个记忆项都成为定点的充要条件。许多模拟实验已证实了这一点。如果定义两个  $N$  维列向量  $\mathbf{X}^{(\alpha)}$  和  $\mathbf{X}^{(\beta)}$  的相关系数如下：

$$R(\mathbf{X}^{(\alpha)}, \mathbf{X}^{(\beta)}) = \frac{[\mathbf{X}^{(\alpha)}]^T \mathbf{X}^{(\beta)}}{\|\mathbf{X}^{(\alpha)}\| \cdot \|\mathbf{X}^{(\beta)}\|}, \quad \|\mathbf{X}^{(\alpha)}\| = \|\mathbf{X}^{(\beta)}\| = N \quad (4-262)$$

当  $N$  为偶数时，式 (4-261) 与下式等效 即

$$R(\mathbf{X}^{(m)}, \mathbf{X}^{(\mu)}) = 0, \quad R(\mathbf{Y}^{(m)}, \mathbf{Y}^{(\mu)}) = 0, \quad \mu, m = 1 \sim M, \quad \mu \neq m \quad (4-263)$$

这表明 当  $\mathbf{W}$  用式 (4-258) 的外积和方法求得时，为了保证  $M$  个记忆项都是 BAM 的定点，任何两个记忆项之间的相关系数必须为 0 即彼此“正交”。在实际应用中达到这一要求是太困难了。更何况即使达到了这一要求，各记忆项周围的吸引域通过模拟实验证实是非常小的。这样 用外积和求  $\mathbf{W}$  的学习算法几乎没有什么实用价值。一种自然思路是改变  $\mathbf{W}$  的学习算法 4.6.2 小节将介绍一种新的有效学习算法。另一种思路是保持  $\mathbf{W}$  的学习算法不变而通过线性变换使正交条件满足 这将在 4.6.3 小节中介绍。

#### 4.6.2 BAM 权矩阵 $\mathbf{W}$ 的有效学习算法

本小节介绍如何设计  $\mathbf{w}$  的学习算法以满足存储  $M$  个记忆项时应该达到的两项要求。下面分别讨论针对每项要求的算法。

##### 1. 为使每个记忆项都成为定点的权矩阵 $\mathbf{W}$ 的学习算法

设有  $M$  个记忆项  $\{\mathbf{X}^{(m)}, \mathbf{Y}^{(m)}\}, m = 1 \sim M$  网络按式 (4-250) 和式 (4-251) 运行。现在讨论如何求  $\mathbf{W}$  使得这  $M$  个记忆项都是定点且可实现一步记忆提取，即当  $\mathbf{X}(0) = \mathbf{X}^{(m)}$ ，则  $\mathbf{Y}(1) = \mathbf{Y}^{(m)}$  或者当  $\mathbf{Y}(0) = \mathbf{Y}^{(m)}$ ，则  $\mathbf{X}(1) = \mathbf{X}^{(m)}, \forall m = 1 \sim M$ 。为了节省篇幅，下面给出的论点不作证明，这些证明可在文献 [90] 中查到。

(1)  $M$  个记忆项都是定点的充分条件是

$$\left\{ \begin{array}{l} \left( \sum_{i=1}^N w_{ij} x_i^{(m)} \right) y_j^{(m)} = \beta_j^{(m)} > 0, \quad j = 1 \sim P, \quad m = 1 \sim M \\ \left( \sum_{j=1}^P w_{ij} y_j^{(m)} \right) x_i^{(m)} = \alpha_i^{(m)} > 0, \quad i = 1 \sim N, \quad m = 1 \sim M \end{array} \right\} \quad (4-264)$$

(2) 如果  $M \leq NP/(N+P)$  且各  $\mathbf{X}^{(m)}, \mathbf{Y}^{(m)}$  分别线性独立，则至少存在一个  $\mathbf{W}$  使得式 (4-264) 成立。该  $\mathbf{W}$  保证所有记忆项都是定点且可实现一步记忆提取。

各  $\mathbf{X}^{(m)}$  线性独立和各  $\mathbf{Y}^{(m)}$  线性独立是一个相当宽松的弱约束条件，在大多数实际情

况中都能得到满足。可以看到，当  $N = P$  时，最大记忆项个数（即绝对记忆容量）为  $M_0 = N/2$ 。考虑到网络含  $2N$  个神经元，则相对记忆容量为  $R = M_0/2N = 0.25$ 。

(3)  $\mathbf{W}$  的批处理学习算法。现在要在 (2) 的约束得到满足的条件下，求  $\mathbf{W}$  使式 (4-264) 成立。为此可以设定一个以  $\mathbf{W}$  为变元的目标函数  $J(\mathbf{W})$  如果有一  $\mathbf{W}$  使  $J(\mathbf{W})$  达到极小值，该  $\mathbf{W}$  即是所需解。这时可以用最陡下降迭代算法来求此解。 $J(\mathbf{W})$  定义如下：

$$J(\mathbf{W}) = \sum_{\substack{i=1 \\ (m,i) \in S_X}}^N \sum_{m=1}^M (-\alpha_i^{(m)}) + \sum_{\substack{j=1 \\ (m,j) \in S_Y}}^P \sum_{m=1}^M (-\beta_j^{(m)}) \quad (4-265)$$

其中  $S_X, S_Y$  的定义是

$$S_X = \{(m, i) \mid \alpha_i^{(m)} \leq 0\} \quad S_Y = \{(m, j) \mid \beta_j^{(m)} \leq 0\} \quad (4-266)$$

这样，当且只有当  $\alpha_i^{(m)} \leq 0$  或  $\beta_j^{(m)} \leq 0$  时，它们才在  $J(\mathbf{W})$  中出现。如果能求得一最佳  $\mathbf{W}$  使所有  $\alpha_i^{(m)} > 0$  且  $\beta_j^{(m)} > 0$ ，这时  $S_X$  和  $S_Y$  为空集， $J(\mathbf{W}) = 0$  即达其最小值：

$$J(\mathbf{W}) = \min_{\mathbf{W}} \{J(\mathbf{W})\} = 0 \quad (4-267)$$

此  $\mathbf{W}$  即为所需的解。为求得  $\mathbf{W}$  从随机初值  $\mathbf{W}(0)$  出发，按节拍  $k = 0, 1, 2, \dots$  进行迭代计算

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}(k) \quad (4-268)$$

$$\Delta w_{ij}(k) = -\eta \left. \frac{\partial J(\mathbf{W})}{\partial w_{ij}} \right|_{\mathbf{W}=\mathbf{W}(k)}, \quad i = 1 \sim N, j = 1 \sim P$$

其中  $\eta > 0$  为步幅。将式 (4-264) 代入 (4-265) 得到

$$\begin{aligned} \left. \frac{\partial J(\mathbf{W})}{\partial w_{ij}} \right|_{\mathbf{W}=\mathbf{W}(k)} &= \sum_{\substack{m=1 \\ (m,i) \in S_X(k)}}^M -\left(\frac{\partial \alpha_i^{(m)}}{\partial w_{ij}}\right) + \sum_{\substack{m=1 \\ (m,j) \in S_Y(k)}}^M \left(-\frac{\partial \beta_j^{(m)}}{\partial w_{ij}}\right) \\ &= \sum_{\substack{m=1 \\ (m,i) \in S_X(k)}}^M (-x_i^{(m)} y_j^{(m)}) + \sum_{\substack{m=1 \\ (m,j) \in S_Y(k)}}^M (-x_i^{(m)} y_j^{(m)}) \end{aligned} \quad (4-269)$$

其中  $S_X(k) = \{(m, i) \mid \alpha_i^{(m)}(k) \leq 0\}$ ,  $S_Y(k) = \{(m, j) \mid \beta_j^{(m)}(k) \leq 0\}$  这样，式 (4-269) 可以写成

$$\left. \frac{\partial J(\mathbf{W})}{\partial w_{ij}} \right|_{\mathbf{W}=\mathbf{W}(k)} = - \sum_{m=1}^M x_i^{(m)} y_j^{(m)} [\varphi(\alpha_i^{(m)}(k)) + \varphi(\beta_j^{(m)}(k))] \quad (4-270)$$

其中函数  $\varphi(\cdot)$  用下式计算：

$$\varphi(u) = \begin{cases} 1, & u \leq 0 \\ 0, & u > 0 \end{cases} \quad (4-271)$$

文献 [91] 证明，当 2) 的约束条件满足时，一定存在一个  $\hat{\mathbf{W}}$  使式 (4-264) 成立，即保证  $J(\mathbf{W}) = 0$ 。这时用上列迭代算法一定能收敛到此  $\mathbf{W}$  上。这是这个算法的突出优点。此外，算法是否收敛极易判断：当且只有当所有  $w_{ij}(k) = 0$  时即达到收敛。最后，此算法步幅可取为  $\eta = 1$ 。

(4)  $\mathbf{W}$  的在线学习算法。在批处理算法中，每一次迭代计算都涉及所有记忆项，而在线学习时，每个迭代节拍  $k$  只依次取出一个记忆项来进行迭代计算。所取出的记忆项用  $\{\mathbf{X}(k), \mathbf{Y}(k)\}$  表示，并且可以用式 (4-264) 计算出相应的  $\alpha_i(k)$  和  $\beta_j(k)$ 。这时迭代计算公式为

$$w_{ij}(k+1) = w_{ij}(k) + \eta x_i(k) y_j(k) [\varphi(\alpha_i(k)) + \varphi(\beta_j(k))].$$

$$i = 1 \sim N, j = 1 \sim P \quad (4-272)$$

其中步幅可取为  $\eta = 1$ 。文献[90]证明,只要符合(2)的条件,此算法必定收敛。在线学习算法的优点是不必一次将全部记忆项采集完备,而可以随采随学,因此可用于实时情况。此外,每个记忆项并非只用一次;而应反复多次使用,直至收敛为止。

## 2. 为使每个记忆项周围的吸引域尽量大的权矩阵 $W$ 学习算法

(1) 如何计算一个 BAM 的吸引域?下面给出计算吸引域大小的一个方法<sup>[90]</sup>。设 BAM 的权矩阵为  $W$  若  $X^{(m)}$  和  $Y^{(m)}$  是记忆项  $X^{(m)}$  和  $Y^{(m)}$  的缺损样本。当取  $X(0) = X^{(m)}$  时  $Y(1) = Y^{(m)}$  且  $X(2) = X^{(m)}$  则称  $\hat{X}^{(m)}$  在  $X^{(m)}$  的一次恢复吸引域中。类似地,当取  $Y(0) = \hat{Y}^{(m)}$  时  $X(1) = X^{(m)}$  且  $Y(2) = Y^{(m)}$  则称  $\hat{Y}^{(m)}$  在  $Y^{(m)}$  的一次恢复吸引域中。设下列条件成立(其中  $\alpha_i^{(m)}, \beta_j^{(m)}$  见式(4-264)):

$$\alpha_i^{(m)} \geq F_X > 0, \quad i = 1 \sim N, \quad \beta_j^{(m)} \geq F_Y > 0, \quad j = 1 \sim P \quad (4-273)$$

且设  $W_{\max} = \max(|w_{ij}|)$ , 那么  $X^{(m)}$  周围吸引半径为  $D_X = F_X / (2W_{\max})$ ,  $Y^{(m)}$  周围吸引半径为  $D_Y = F_Y / (2W_{\max})$ 。即当  $X^{(m)}$  或  $Y^{(m)}$  满足下列条件时,可实现一次恢复吸引,

$$d_H(\hat{X}^{(m)}, X^{(m)}) \leq D_X, \quad d_H(\hat{Y}^{(m)}, Y^{(m)}) \leq D_Y \quad (4-274)$$

其中  $d_H(\cdot, \cdot)$  为汉明距离 见式(4-259)。如果对于任何  $m = 1 \sim M$ , 式(4-273) 皆成立, 那么每个记忆项周围的吸引半径都是  $D_X, D_Y$ 。

(2) 为使吸引域达到最大的权矩阵  $W$  在线学习算法。基于(1)的吸引域计算方法, 可以构成如下学习算法。设有  $M$  个记忆项  $\{X^{(m)}, Y^{(m)}\}, m = 1 \sim M$ 。由(1)中计算吸引半径所需的  $F_X, F_Y$  和  $W_{\max}$  诸参数不可能预先知道, 所以可首先设定  $F_X = F_Y = 0, W_{\max}$  设定为某个适当的正数。这样, 在设置了随机权初值  $W(0)$  后, 即可按节拍  $k = 0, 1, 2, \dots$  进行学习。对于每一节拍  $k$  依次取一个记忆项 记为  $\{X(k), Y(k)\}$  (当所有记忆项都取过后再从头取)接着按式(4-264)计算出  $\alpha_i(k), i = 1 \sim N; \beta_j(k), j = 1 \sim P$ 。即  $\alpha_i(k) = \left( \sum_{j=1}^P w_{ij}(k) y_j(k) \right) x_i(k), \beta_j(k) = \left( \sum_{i=1}^N w_{ij}(k) x_i(k) \right) y_j(k)$ 。权调整公式为

$$w_{ij}(k+1) = G[w_{ij}(k) + \Delta w_{ij}(k)], \quad i = 1 \sim N, j = 1 \sim P \left\{ \begin{array}{l} \Delta w_{ij}(k) = \eta x_i(k) y_j(k) [\varphi(\alpha_i(k) - F_X) + \varphi(\beta_j(k) - F_Y)] \end{array} \right\} \quad (4-275)$$

其中  $G[\cdot]$  用式(4-245) 计算  $\varphi(\cdot)$  用式(4-271) 计算。 $\eta$  是一个可选择的步幅,  $0 < \eta < 1$ 。式中函数  $\varphi(\cdot)$  的作用是当  $\alpha_i(k) \leq F_X$  或  $\beta_j(k) \leq F_Y$  时才对相应的权进行调整, 否则  $w_{ij}$  保持不变。函数  $G[\cdot]$  是保证各  $w_{ij}$  的最大绝对值不超过  $W_{\max}$ 。如何选择  $W_{\max}$  和  $\eta$  可参阅文献[90]、[77]。如果存在  $W$  使得式(4-273) 成立, 那么此迭代算法必然收敛到这个  $W$  上。如果迭代计算收敛, 则可以将  $F_X, F_Y$  稍微增大并再次进行迭代计算。此过程可继续进行直到算法不收敛为止, 从而可以求得  $F_X, F_Y$  的最大值。

模拟实验表明<sup>[90]</sup>, 这个学习算法在增大记忆容量、扩张吸引域以及改进相关记忆项的存储效果等诸方面都比传统方法有明显改善。例如, 用  $N = P = 50$  的 BAM 进行实验, 记忆项为相互独立且其中每个分量取 1 或 -1 的概率都是 1/2 令  $M$  在 1 ~ 25 范围内变

化。在实验中用了 100 组记忆样本来统计记忆项成为定点的个数占总记忆项个数之百分比。实验结果表明 用外积和算法时 当  $M \leq 3$  此比值为 1 当  $M = 5$  比值为 0.95;  $M = 7$ , 比值为 0.42;  $M = 9$  比值为 0.06。而用此改进算法时 当  $M \leq 23$  比值为 1 当  $M = 25$  比值为 0.87。可以看到 后者的相对记忆容量  $R$  非常接近于前面推断的 0.25 这一值。而采用外积和方法时  $R$  只能接近于 0.05。

BAM 的应用包括模式识别<sup>[92]</sup>、图像处理<sup>[93]</sup>、控制系统的故障处理<sup>[94]</sup> 文献[95] 给出了 BAM 应用的综述。

#### 4.6.3 通过线性变换改善 BAM 的性能

前文已指出, 如果  $M$  个存储项中各对存储项两两正交 (见式 4-263)) 则按照外积和的方法求得权矩阵  $W$  并按式 (4-254)、(4-255) 运行时 各存储记忆项都是定点。如果此要求不满足 可以设想采取下列线性变换 压扩与移位):

$$\tilde{x}_i = \varphi_i x_i - f_i, \quad i = 1 \sim N, \quad \tilde{y}_j = \gamma_j y_j - g_j, \quad j = 1 \sim P \quad (4-276)$$

变换后的记忆项将是

$$\tilde{\mathbf{X}}^{(m)} = [\tilde{x}_1^{(m)}, \dots, \tilde{x}_N^{(m)}]^T, \quad \tilde{\mathbf{Y}}^{(m)} = [\tilde{y}_1^{(m)}, \dots, \tilde{y}_P^{(m)}]^T, \quad m = 1 \sim M$$

它们两两之间的相关系数将随  $\varphi_i, f_i, \gamma_j, g_j$  而变化。根据式 4-262) 变换后各对记忆项相关系数绝对值之和将等于

$$\begin{aligned} \sum_{\substack{m, \mu=1 \\ m \neq \mu}}^M |R(\tilde{\mathbf{X}}^{(m)}, \tilde{\mathbf{X}}^{(\mu)})| &= \sum_{\substack{m, \mu=1 \\ m \neq \mu}}^M \frac{|[\tilde{\mathbf{X}}^{(m)}]^T \tilde{\mathbf{X}}^{(\mu)}|}{\|\tilde{\mathbf{X}}^{(m)}\| \cdot \|\tilde{\mathbf{X}}^{(\mu)}\|} \\ &= \sum_{\substack{m, \mu=1 \\ m \neq \mu}}^M \frac{\left| \left[ \sum_{i=1}^N (\varphi_i x_i^{(m)} - f_i)(\varphi_i x_i^{(\mu)} - f_i) \right] \right|}{\left[ \sum_{i=1}^N (\varphi_i x_i^{(m)} - f_i)^2 \sum_{i=1}^N (\varphi_i x_i^{(\mu)} - f_i)^2 \right]^{\frac{1}{2}}}, \\ \sum_{\substack{m, \mu=1 \\ m \neq \mu}}^M |R(\tilde{\mathbf{Y}}^{(m)}, \tilde{\mathbf{Y}}^{(\mu)})| &= \sum_{\substack{m, \mu=1 \\ m \neq \mu}}^M \frac{|[\tilde{\mathbf{Y}}^{(m)}]^T \tilde{\mathbf{Y}}^{(\mu)}|}{\|\tilde{\mathbf{Y}}^{(m)}\| \cdot \|\tilde{\mathbf{Y}}^{(\mu)}\|} \\ &= \sum_{\substack{m, \mu=1 \\ m \neq \mu}}^M \frac{\left| \left[ \sum_{j=1}^P (\gamma_j y_j^{(m)} - g_j)(\gamma_j y_j^{(\mu)} - g_j) \right] \right|}{\left[ \sum_{j=1}^P (\gamma_j y_j^{(m)} - g_j)^2 \sum_{j=1}^P (\gamma_j y_j^{(\mu)} - g_j)^2 \right]^{\frac{1}{2}}} \quad (4-277) \end{aligned}$$

如果各记忆项本来就两两正交, 这时可取各变换系数为  $\varphi_i = 1, f_i = 0, \gamma_j = 1, g_j = 0, \forall i, j$ 。这样 即得原来的 BAM 且上述两个相关系数绝对值之和等于 0。如果各记忆项并非两两正交, 就应该求诸  $\varphi_i, f_i, \gamma_j, g_j$ , 使得这两个和式尽量接近于 0 即各对  $\tilde{\mathbf{X}}^{(m)}, \tilde{\mathbf{X}}^{(\mu)}$  和  $\tilde{\mathbf{Y}}^{(m)}, \tilde{\mathbf{Y}}^{(\mu)}$  之间接近于两两正交。这样 BAM 的运行方程可以作相应修正, 例如式 (4-250) 和式 (4-251) 中的调整变化部分应该改成为

$$\begin{aligned} y_j(k+1) &= \operatorname{sgn} \left[ \sum_{i=1}^N w_{ij} (\varphi_i x_i - f_i) \right], \quad j = 1 \sim P \text{ 及 } x_i(k+1) \\ &= \operatorname{sgn} \left[ \sum_{j=1}^P w_{ij} (\gamma_j y_j - g_j) \right], \quad i = 1 \sim N \end{aligned} \quad (4-278)$$

而权矩阵  $\mathbf{W}$  相应地可用下式计算：

$$\mathbf{W} = \sum_{m=1}^M [\tilde{\mathbf{X}}^{(m)}]^T \tilde{\mathbf{Y}}^{(m)} \quad (4-279)$$

此算法的关键在于，如何求得各  $\varphi_i, f_i, \gamma_j, g_j$  使得式 (4-277) 的两个相关系数绝对值之和达到尽可能小的值，文献[96] 通过启发式的讨论提出求这些系数的如下方案：

$$\left. \begin{aligned} f_i &= \lambda \sum_{m=1}^M x_i^{(m)}, \quad \varphi_i = [1 + f_i]^{\frac{1}{2}}, \quad i = 1 \sim N \\ g_j &= \mu \sum_{m=1}^M y_j^{(m)}, \quad \gamma_j = [1 + g_j]^{\frac{1}{2}}, \quad j = 1 \sim P \end{aligned} \right\} \quad (4-280)$$

其中  $\lambda, \mu$  是两个待定的参数。模拟实验表明<sup>[96]</sup> 当  $N = P = 20$  且输入各记忆项中度及高度相关时，取  $\lambda = \mu = 0.06 \sim 0.08$  较之取  $\lambda = \mu = 0$  的存储效果有明显改善（后者为标准 BAM 情况）。事实上如何求各参数的问题仍是未解决。

最近的一篇文献还提出用着色图的思路来求  $\mathbf{W}$  以改善存储效果<sup>[97]</sup> 可以参考。

#### 4.6.4 BSB 神经网络

BSB 是 brain-state-in-a-box 的缩写。这种网络由  $N$  个神经元构成，每个神经元的输出  $x_i$  局限在  $[-1, 1]$  范围内 因此由  $N$  个输出构成的状态向量（简称状态） $\mathbf{X} = [x_1, x_2, \dots, x_N]^T$  被局限在一个  $N$  维超立方格之中。此超立方格犹如一个盒子，所以这种神经网络模型称为“盒中脑状态”模型 即 BSB。网络的运行规则如下 按照离散时间节拍  $k$  当  $k$  时刻网络状态为  $\mathbf{X}(k)$  则  $k+1$  时刻网络状态  $\mathbf{X}(k+1)$  可用下式计算：

$$\mathbf{X}(k+1) = \mathbf{g}[\mathbf{X}(k) + \alpha \mathbf{W} \mathbf{X}(k)] = \mathbf{g}[(\mathbf{I}_N + \alpha \mathbf{W}) \mathbf{X}(k)] \quad (4-281)$$

其中  $\mathbf{g}[\cdot]$  运算的定义是，若  $\mathbf{U} = [u_1, \dots, u_N]^T$  则

$$\mathbf{g}[\mathbf{U}] = [g(u_1), \dots, g(u_N)]^T, \quad g(u_i) = \begin{cases} 1, & u_i \geq 1 \\ u_i, & -1 < u_i < 1 \\ -1, & u_i \leq -1 \end{cases} \quad (4-282)$$

$\mathbf{W}$  是一个对称实矩阵。 $\alpha$  是一个适当选择的步幅， $\alpha > 0$ 。 $\mathbf{I}_N$  是一个  $N$  维单位阵。这种网络是由 J. A. Anderson 等在 1977 年作为“感知分类器”提出的<sup>[98]</sup>，此后吸引了若干研究者的兴趣<sup>[99,100]</sup> 并且与“细胞神经网络”有密切关系<sup>[101]</sup>。后来更扩展为广义 BSB(GBSB)<sup>[102]</sup>，其运行方程为

$$\mathbf{X}(k+1) = \mathbf{g}[(\mathbf{I}_N + \alpha \mathbf{W}) \mathbf{X}(k) + \alpha \mathbf{b}] \quad (4-283)$$

其中  $\mathbf{b}$  是一个  $N$  维实向量  $\mathbf{W}$  不限定为对称阵。BSB 或 GBSB 的最重要应用目标是联想记忆，即如何设计和计算  $\mathbf{W}, \mathbf{b}$  和  $\alpha$  以使得所有记忆项都成为定点且吸引域足够大。迄今为止在这个研究方向上已取得了一些进展，见文献[103]~[105]。但是这种网络的存储容量、吸引域的计算方法等方面仍没有系统有效的计算方法。

## 参考文献

- [1] Hopfield J J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Nat. Acad. Sci. USA*, 1982, 79: 2554 ~ 2558
- [2] Hopfield J J. Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Nat. Acad. Sci. USA*, 1984, 81: 3088 ~ 3092
- [3] Hopfield J J. Tank D. Neural computation of decisions in optimization problems. *Bio. Cybern.*, 1985, 52: 141 ~ 152
- [4] Takahashi Yoshikane. Mathematical improvement of the Hopfield model for feasible solutions to the traveling salesman problem by a synapse dynamical system. *IEEE Trans. on SMAC-Pt. B: Cybernetics*, Dec. 1998, 28(6): 906 ~ 919
- [5] Smith K. An argument for abandoning the traveling salesman problem as a neural network benchmark. *IEEE Trans. on Neural Networks*, Nov. 1996, 7(6): 1542 ~ 1544
- [6] 徐利治等. 现代数学手册. 见 经济数学卷第 9 篇 —— 整数规划. 武汉: 华中科技大学出版社, 2001
- [7] Smith K, et al. Static and dynamic channel assignment using neural networks. *IEEE Journal on selected areas in communication*, 1997, 15(2): 238 ~ 249
- [8] Hopfield J J. Tank D W. Computing with neural circuits: a model. *Science*, 1986, 233: 625 ~ 633
- [9] Abe S. Theories on the Hopfield neural networks. In: *Proc. of International Joint Conference on Neural Networks*. Washington D. C.: 1989, 1: 557 ~ 564
- [10] Abe S, et al. Solving inequality constrained combinatorial optimization problems by the Hopfield neural networks. *Neural Networks*, 1992, 5: 663 ~ 670
- [11] Durbin R, et al. An analysis of the elastic net approach to the traveling salesman problem. *Neural Computation*, 1989, 1: 348 ~ 358
- [12] Durbin R, Willshaw D. An analogue approach to the traveling salesman problem. *Nature*, 1987, 326: 689 ~ 691
- [13] Hulle M M Van. A goal programming network for linear programming. *Biol. Cybern.*, 1991, 65: 243 ~ 252
- [14] Yuille A L. Generalized deformable models, statistical physics and matching problems. *Neural Computation*, 1990, 2: 1 ~ 24
- [15] Angeniol B, et al. Self-organizing feature maps and the traveling salesman problem. *Neural Networks*, 1988, 1: 289 ~ 293
- [16] Budinich M. A self-organizing neural network for the traveling salesman problem that is competitive with simulated annealing. In: *Proc. ICANN. Sorrento. Italy; May, 1994*, 1: 359 ~ 361
- [17] Favata F, Walker R. A study of the application of Kohonen-type neural networks to traveling salesman problem. *Biol. Cybern.*, 1991, 64: 463 ~ 468
- [18] Fort J. Solving a combinatorial problem via self-organizing process: An application of the Kohonen algorithm to the traveling salesman problem. *Biol. Cybern.*, 1988, 59: 34 ~ 40
- [19] Smith K, et al. Neural techniques for combinatorial optimization with applications. *IEEE Trans. on Neural Networks*, Nov. 1998, 9(6): 1301 ~ 1318
- [20] Banzhaf W. The “molecular” traveling salesman. *Biol. Cybern.*, 1990, 64: 7 ~ 14



- [21] Fogel D B. An evolutionary approach to the traveling salesman problem. *Biol. Cybern.* , 1989, 60: 139 ~ 144
- [22] Wilson G V, Pawley G S. On the stability of the traveling salesman problem algorithm of Hopfield and Tank. *Biol. Cybern.* , 1988, 58: 63 ~ 70
- [23] Peterson Carsten. Parallel distributed approaches to combinatorial optimization: benchmark studies on traveling salesman problem. *Neural Computation* 2, 1990: 261 ~ 269
- [24] Aiyer S V B, et al. A theoretical investigation into the performance of the Hopfield model. *IEEE Trans. on Neural Networks*, June 1990, 1(2)
- [25] Tagliarini G A. Optimization using neural networks. *IEEE Trans. on Computer*, 1991, 40(12): 1347 ~ 1358
- [26] Gee A H, Prager R W. Polyhedral combinatorics and neural networks. *Neural Computation*, 1994, 6: 161 ~ 180
- [27] Tank D W, Hopfield J J. Simple neural optimization networks; an A/D converter, signal decision circuit, and a linear programming circuit. *IEEE Trans. CAS*, May 1986, 33(5): 533 ~ 541
- [28] Lee B W, Sheu B J. Design of neural-based A/D converter using modified Hopfield network. *IEEE J. of Solid-State Circuits*, 1989, 24(4): 1129 ~ 1135
- [29] Takeda Mitsuo, Goodman Joseph W. Neural networks for computation: number representations and programming complexity. *Applied Optics*, 1986, 25(18): 3033 ~ 3046
- [30] Hackman S T, et al. Fast, effective algorithms for simple assembly line balancing problems. *Operations Res.* , 1989, 37(6): 916 ~ 924
- [31] McCormick S T, et al. Sequencing in an assembly line with blocking to minimize cycle time. *Operations Res.* , 1989, 37(6): 925 ~ 935
- [32] Sherali H D, Brown E L. A quadratic partial assignment and packing model and algorithm for the airline gate assignment problem. In: P. M. Pardalos, et al, ed. *Quadratic Assignment and Related Problems*. Providence (RI): American Math. Soci. , 1993
- [33] Smith K, et al. Traditional heuristic versus Hopfield neural-network approaches to a car sequencing problem. *European Operational Res.* , 1996, 93(2): 300 ~ 316
- [34] O'Kelly M E. A quadratic integer program for the location of interacting hub facilities. *European J. Operational Res.* , 1987, 32: 393 ~ 404
- [35] Duque M, et al. Static and dynamic channel assignment using simulated annealing. In: B. Yuhas, et al, ed. *Neural Networks in Telecommunications*. Boston (MA): Kluwer, 1994
- [36] Funabiki N, Takefuji Y. A neural network parallel algorithm for channel assignment problems in cellular radio networks. *IEEE Trans. Veh. Technol.* , 1992, 41(4): 430 ~ 437
- [37] Kunz D. Channel assignment for cellular radio using neural networks. *IEEE Trans. Veh. Technol.* , 1991, 40(1): 188 ~ 193
- [38] Kohonen T. Self-organized formation of topologically correct feature maps. *Biol. Cybern.* , 1982, 43: 59 ~ 69
- [39] Garey M R, Johnson D S. *Computers and intractability—a guide to the theory of NP-completeness*. Freeman W H, San Francisco: 1979
- [40] Schrijver A. *Theory of linear and integer programming*. Chichester: John Wiley, 1986
- [41] Peterson C, Anderson J R. Neural networks and NP-complete optimization problems: a performance study on the graph bisection problem. *Complex Syst.* , 1988, 2(1): 59 ~ 89

- [42] Peterson C, Söderberg B. A new method for mapping optimization problems onto neural networks. *Int. J. Neural Syst.*, 1989, 1(1): 3 ~ 22
- [43] Bout D E Van den, Miller T K III. Graph partitioning using annealed neural networks. *IEEE Trans. on Neural Networks*, 1990, 1(2): 192 ~ 203
- [44] Langevin A, et al. Classification of travelling salesman problem for mutations. *Oper. Res. Lett.*, 1990, 9(2): 127 ~ 132
- [45] Gall A Le, Zissimopoulos V. *Extended Hopfield models for combinatorial optimization*. *IEEE Trans. on Neural Networks*, Jan. 1999, 10(1): 72 ~ 80
- [46] Ohlsson M, et al. Neural networks for optimization problems with inequality constraints; the knapsack problem. *Neural Computation*, 1993, 5: 331 ~ 339
- [47] Ohlsson M, Pi Hong. A study of the mean field approach to knapsack problems. *Neural Networks*, 1997, 10(2): 263 ~ 271
- [48] Press W P, et al. *Numerical recipes the art of scientific computing*. Cambridge: Cambridge University Press, 1986
- [49] Ansari N, et al. A new method to optimize the satellite broadcasting schedules using the mean field annealing of a Hopfield neural network. *IEEE Trans. on Neural Networks*, March 1995, 6(2): 470 ~ 483
- [50] Morris R J T, Samadi B. Neural network control of communication systems. *IEEE Trans. on Neural Networks*, July 1994, 5(4): 639 ~ 650
- [51] Ali M K M, Kamoun F. Neural networks for shortest path computation and routing in computer networks. *IEEE Trans. on Neural Networks*, Nov. 1993 4(6): 941 ~ 954
- [52] Gelenbe E, et al. *Improved neural heuristics for multicasting routing*. *IEEE J. on Selected Areas in Commun.*, Feb. 1997, 15(2): 147 ~ 155
- [53] Park Y-K, Lee G. Applications of neural networks in high-speed communication networks. *IEEE Commun. Mag.*, Oct. 1995, 33(10): 68 ~ 74
- [54] Tarraf Ahmed A, et al. Intelligent traffic control for ATM broadband networks. *IEEE Commun. Mag.*, Oct. 1995, 33(10): 76 ~ 82
- [55] Gelenbe E. Scanning the issue—special issue on engineering applications of artificial neural networks. *Proc. IEEE*, Oct. 1996, 84(10): 1 ~ 2
- [56] Habib I W. Applications of neural computing in traffic management of ATM networks. *Proc. IEEE*, Oct. 1996, 84(10): 1430 ~ 1441
- [57] Hiramatsu A. *ATM communication network control by neural networks*. *IEEE Trans. on Neural Networks*, Mar. 1990, 1(1): 122 ~ 130
- [58] Ambrose Barry E, Googman Rodney M. Neural networks applied to traffic management in telephone networks. *Proc. IEEE*, Oct. 1996, 84(10): 1421 ~ 1429
- [59] Kechriotis George I, Manolakos Elias S. Hopfield neural network implementation of the optimal CDMA multiuser detector. *IEEE Trans. on Neural Networks*, Jan. 1996, 7(1): 131 ~ 141
- [60] Shen Dinggang, Horace H S Ip. A Hopfield neural network for adaptive image segmentation; an active surface paradigm. *Pattern Recognition Letters*, 1997, 18(1): 37 ~ 48
- [61] Srinivasan A, Batur C. Hopfield/ART-1 neural network-based fault detection and isolation. *IEEE Trans. on Neural Networks*, Nov. 1994, 5(6): 890 ~ 899
- [62] Cavalieri S, Mirabella O. Neural networks for process scheduling in real-time communication systems.

- IEEE Trans. on Neural Networks, Sep. 1996, 7(5): 1272 ~ 1285
- [63] Bharitkar Sunil, Mendel Jerry M. The hysteretic Hopfield neural network. IEEE Trans. on Neural Networks, July 2000 . 11(4): 879 ~ 888
  - [64] Martin-Valdiva M, et al. Improving local minima of Hopfield networks with augmented Lagrange multipliers for large scale TSPs. Neural Networks, 2000, 13 . 283 ~ 285
  - [65] Hebb D O. The organization of behavior. New York: Wiley, 1949
  - [66] Blum E K. Mathematical aspects of outer-product asynchronous content-addressable memories. Biol. Cybern. . 1990, 62 : 337 ~ 348
  - [67] 杨行峻 郑君里. 人工神经网络. 北京: 高教出版社, 1992 第三章
  - [68] McEliece R J, et al. The capacity of Hopfield associative memory. IEEE Trans. on Information Theory, July 1987. 33(4): 461 ~ 482
  - [69] Morita M. Associative memory with nonmonotone dynamics. Neural Networks, 1993, 6(1) : 115 ~ 126
  - [70] Yanai H-F, Amari S. Auto-associative memory with two-stage dynamics of nonmonotonic neurons. IEEE Trans. on Neural Networks, July 1996. 7(4). 803 ~ 815
  - [71] 格涅坚科 Б.Б. 概率论教程. 北京: 高等教育出版社, 1956
  - [72] Abbott L F, Kepler T B. Optimal learning in neural network memories. Journal of Physics A: Mathematical and General, 1989 . 22: 711 ~ 717
  - [73] Diederich S, Oppen M. Learning of correlated patterns in spin-glass networks by local learning rules. Physical Review Letters, 1987. 58. 949 ~ 952
  - [74] Krauth W, Mezard M. Learning algorithms with optimal stability in neural networks. Journal of Physics A: Mathematical and General, 1987 . 20: 745 ~ 752
  - [75] Gardner E. The space of interactions in neural network models. Journal of Physics A: Mathematical and General, 1988, 21: 257 ~ 270
  - [76] Cover T M. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. IEEE Trans. on Electronic Computers, 1965. 14: 326 ~ 334
  - [77] Wang Tao. Improving recall in associative memories by dynamic threshold. Neural Networks, 1994. 7(9): 1379 ~ 1385
  - [78] Storkey A J. Increasing the capacity of the Hopfield network without sacrificing functionality. In: Gerstnev W, et al, ed. ICANN97: Lecture Notes in Computer Science 1327. Berlin: Springer. 451 ~ 456
  - [79] Storkey A J, Valabregue R. The basin of attraction of a new Hopfield learning rule. Neural Networks, 1999. 12: 869 ~ 876
  - [80] Storkey A J. Valabregue R. Hopfield learning rule with high capacity storage of time-correlated patterns. Electronic Letters, Oct. 1997. 33(21): 1803 ~ 1804
  - [81] Amari S I, Maginu K. Statistical neurodynamics of associative memory. Neural Networks, 1988, 1(1): 63 ~ 73
  - [82] Chang J Y, Wu C C. Design of Hopfield-type associative memory with maximal basin of attraction. Electronic Letters, 1993. 29(24): 2128 ~ 2130
  - [83] Forrest B M. Content-addressability and learning in neural networks. Journal of Physics A: Mathematical and General, 1988, 21: 245 ~ 255
  - [84] Horner H, et al. Transients and basins of attraction in neural network models. Zeitschrift für

Physik B, 1989, 76: 381 ~ 398

- [85] Viana L, Coolen A. Attraction domains in neural networks. *Journal de Physique*, 1993, 1(3): 777 ~ 786
- [86] Kanter I, Sompolinsky H. Associative recall of memory without errors. *Physical Review A: General Physics*, 1987, 35(1): 380 ~ 392
- [87] Kosko B. Bidirectional associative memories. *IEEE Trans. on Syst. Man. Cybern.*, 1988 18(1): 49 ~ 60
- [88] Kosko B. Adaptive bidirectional associative memories. *Appl. Optics.*, 1987, 26: 4947 ~ 4960
- [89] Zhang B-L, et al. Performance analysis of the bidirectional associative memory and an improved model from the matched-filtering viewpoint. *IEEE Trans. on Neural Networks*, 1993, 4(5): 864 ~ 872
- [90] Wang Tao, et al. Design bidirectional associative memories with optimal stability. *IEEE Trans. on Syst., Man, and Cybernetics*, May 1994, 24(5): 778 ~ 790
- [91] Duda R O, Hart P E. Pattern classification and scene analysis. New York: Wiley, 1973
- [92] Wang Y F, et al. Two coding strategies for bidirectional associative memory. *IEEE Trans. on Neural Networks*, 1990, 1(1): 81 ~ 92
- [93] Dunning G, et al. Optical Holographic associative memory using a phase conjugate resonator. *Proc. of the SPIE*, 1986, 625: 205 ~ 213
- [94] Bavarian B. Introduction to neural networks for intelligent control. *IEEE Control Systems Magazine*, April 1988; 3 ~ 7
- [95] Simpson P K. Artificial neural systems: foundations, paradigms, applications and implementations. Elmsford: Pergamon Press, 1990
- [96] Lenze Burkhard. Complexity preserving of the capacity of bidirectional associative memories by dilation and translation. *Neural Networks*, 1998, 11: 1041 ~ 1048
- [97] Giménez-Martínez V. A modified Hopfield auto-associative memory with improved memory. *IEEE Trans. on Neural Networks*, 2000, 11(4): 867 ~ 878
- [98] Anderson J A, et al. Distinctive features, categorical perception, and probability learning: some applications of neural model. In: Anderson J A, and Rosenfeld E, ed. *Neurocomputing: Foundations of Research*, Cambridge, MA: MIT Press, 1988
- [99] Greenberg H J. Equilibria of the brain-state-in-a-box (BSB) neural model. *Neural Networks*, 1988, 1(4): 323 ~ 324
- [100] Michel A N, Farrell J A. Associative memories via artificial neural networks. *IEEE Contr. Syst. Mag.*, 1990, 10(3): 6 ~ 17
- [101] Chua L O, Yang L. Cellular neural networks; theory. *IEEE Trans. on Circuits Syst.*, 1988, 35: 1257 ~ 1272
- [102] Lillo W E, et al. Synthesis of brain-state-in-a-box (BSB) based associative memories. *IEEE Trans. on Neural Networks*, 1994, 5(5): 730 ~ 737
- [103] Hui S, Zak S H. Dynamic analysis of the brain-state-in-a-box (BSB) neural models. *IEEE Trans. on Neural Networks*, 1992, 3(1): 86 ~ 94
- [104] Zak S H, et al. Learning and forgetting in generalized brain-state-in-a-box (BSB) neural associative memories. *Neural Networks*, July, 1996, 9(5): 845 ~ 854
- [105] Chan H Y, Zak S H. On neural networks that design neural associative memories. *IEEE Trans.*

on Neural Networks, Mar. 1997 . 8(2): 360 ~ 372

- [106] 孙守宇, 郑君里. Hopfield 网络求解 TSP 的改进算法和理论证明. 电子学报, 1995. 1, 23(1): 73 ~ 78
- [107] 孙守宇, 郑君里. 一种基于神经网络进行通信网划分设计的并行算法. 电子学报, 1994. 10, 22(10): 81 ~ 84
- [108] 孙守宇, 郑君里. 一种改进的 Hopfield 网络算法用于单元布局. 清华大学学报, 1996. 5, 36(5): 7 ~ 11
- [109] 张宇, 郑君里. 基于 ANN 的双重选路算法实现 ATM 路由选择. 电子学报, 1998. 8, 26(8): 139 ~ 142
- [110] 张宇, 郑君里. 利用神经网络优化算法改善分组交换流量控制. 清华大学学报, 1996. 5, 35(53): 32 ~ 37

# 第 5 章 模糊神经网络

## 5.1 概 述

模糊集理论是在总结和概括人类用语言作为媒体进行各种智能活动的基础上，建立的一套数学理论。反映主客观世界的人类语言陈述大多具有模糊性，例如“今天气温很高”所表述的温度并不确定，相反，“今天气温为  $27^{\circ}\text{C}$ ”所述的是一个确定温度。后者称为一个确定数 (crisp number)，前者称为一个模糊数 (fuzzy number)。确定数只有一个取值，而模糊数的取值可能有多个或一个区间。一个模糊数  $A$  可以用某个取值  $x$  隶属于  $A$  的程度  $\mu_A(x)$  来描述， $\mu_A(x)$  称为隶属函数 (membership function)。  $0 \leq \mu_A(x) \leq 1$ ， $\mu_A(x) = 1$  表示  $x$  完全隶属于  $A$ ， $\mu_A(x) = 0$  表示  $x$  不隶属于  $A$ 。所有使  $\mu_A(x)$  取非零值的  $x$  构成的集合称为  $A$  的支 (support)，用  $S_A$  表示，即  $S_A = \{x \mid \mu_A(x) > 0\}$ 。在模糊神经网络中常用的隶属函数为梯形、三角形、钟形和高斯形。以气温为例，设用模糊数  $A_1$ 、 $A_2$ 、 $A_3$  分别表示气温“很高”、“偏高”、“适中”。任一温度值  $x$  对它们的隶属程度为  $\mu_{A_1}(x)$ 、 $\mu_{A_2}(x)$ 、 $\mu_{A_3}(x)$ 。图 5.1 所示是取梯形形式的这组隶属函数的样例。它们的支分别是  $S_{A_1} = \{x \in [32.5, 40]\}$ ， $S_{A_2} = \{x \in [25, 35]\}$ ， $S_{A_3} = \{x \in [10, 30]\}$ 。

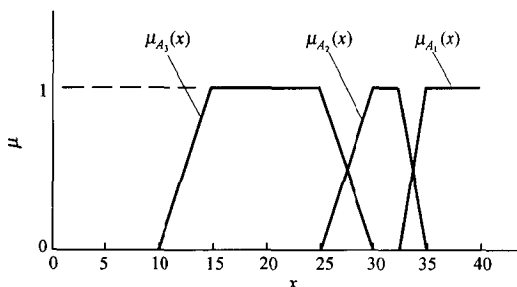


图 5-1 梯形隶属函数

智能信号处理的大多数领域都涉及推理问题，即由一定的前提导出最恰当的结论，其关键是开发学习算法来建立所需的推理规则。这是人类和机器智能系统共同面临的学习问题。模糊逻辑推理系统在处理不确定、复杂、高度非线性、随机、时变系统所产生的信息时远优于人工智能领域中的专家系统，更由于它在不同层次上模仿了人类的智能活动，因此 20 世纪 80 年代以来日益受到计算智能学界的重视。一个多前提 ( $N$  个前提) 单结论的模糊逻辑推理规则可以取“与”及“或”两种形式，取“与”时可表示为

$$\text{IF } x_1 \text{ is } A_1 \text{ and } x_2 \text{ is } A_2 \text{ and } \cdots x_N \text{ is } A_N \text{ THEN } y \text{ is } B \quad (5-1)$$

取“或”时的表示式见下文的式(5-2)。其中  $A_1 \sim A_N$  是  $N$  个前提模糊数,  $x_1 \sim x_N$  是一组前提取值;  $B$  是结论模糊数,  $y$  是结论取值。如何用此规则来进行运算, 如何建立规则, 如何处理多条规则的情况以及多个输出的情况, 将在 5.2 节中讨论。

早期的模糊逻辑推理系统依赖于专家的经验, 用手工方式建立所需的规则。这种方法效率低且难以达到最优化, 尤其当前提数量很多且待完成的推理十分复杂时更难奏效。与人工神经网络相结合并且采取行之有效的非手工学习算法的问题便自然置于研究者的面前。人工神经网络(包括 MLFN, ART, SOM 等)能够实现极复杂的非线性映射且具有很强的学习能力, 但是长期以来存在两个为人诟病的缺点。其一是学习速度慢且可能收敛于目标函数的某个质量不高的局部最优点上; 其二, 网络是一个“黑盒子”, 其中各神经元和权不能赋予明确的物理意义, 这样就不能充分利用已有的知识来改善网络的结构设计以及加快学习速度并避免低质局部最优点。将模糊逻辑(FL)嵌入各种类型的神经网络形成模糊神经网络(FNN)正可以取长补短, 充分发挥二者优点。因此, FNN 自 20 世纪 90 年代初期以来一直是一个研究热点, 至今方兴未艾。

FNN 具有多种型式。按照应用领域来划分, 可以分成用于分类(classification)和模式识别的 FNN、用于聚类(clustering)的 FNN、用于函数逼近的 FNN、用于实现模糊推理的 FNN 等。按照所结合的神经网络类型可以分成基于 ART 的 Fuzzy ART 型 FNN(在 5.4 节中讨论)、基于 TSK 模型的 FNN(在 5.3 节中讨论)、基于 SOM 的 FNN(在 5.5 节中讨论)以及其他类型(在 5.11 节中讨论)。按具体实用目标而言, 又可以分成在系统辨识和控制、时间序列预测、图像和语音识别、雷达和水声、医学、金融、地学、核工业、自然科学等等不同学科中的应用。应该指出, 早期的模糊逻辑控制器(FLC)只是针对一些消费电子产品(洗衣机、空调、冰箱等)而设计的简单操作系统, 而近代 FNN 所面临的问题无论是规模还是复杂性都远远超出了这些简单系统的要求。这样, 依靠专家手工操作来设计 FLC 及其他模糊逻辑系统的简单方法必须用一种系统化的、有坚实理论基础的自动设计方法替代, 从而摆脱尝试一失败一再尝试的局面。

各种类型的 FNN 的学习算法共同方面是都包括结构学习和参数学习两部分。结构学习的目标是按照一定的性能要求, 确定一个模糊逻辑系统包含多少条推理规则(每条规则用下文式(5-2)表示)、每条规则中初步确定的前提和结论(即各前提和结论模糊数的隶属函数)以及由模糊数通过去模糊化得到确定数的方法等。参数学习则是进一步精细调节各隶属函数的参数以及模糊推理规则的其他参数, 使系统臻于最优。各种 FNN 用于不同领域和具体场合时又具有相当大的差异, 本章将分别介绍 FNN 在控制(5.6 节)、时间序列预测(5.7 节)、分类和模式识别(5.8 节)、聚类(5.9 节)等不同领域中的应用特点。

## 5.2 FNN 的结构和类型

### 5.2.1 模糊逻辑推理系统的基本结构

设系统的输入是  $N$  维确定向量  $\mathbf{X} = [x_1, \cdots, x_N]$ , 输出是  $M$  维确定向量  $\mathbf{Y} = [y_1, \cdots, y_M]$ 。系统由模糊化、模糊推理和去模糊化三部分组成, 如图 5-2 所示。输入向量空间用  $q$

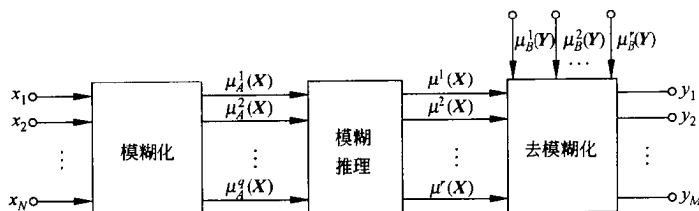


图 5-2 模糊逻辑推理系统结构

个模糊数  $A^1 \sim A^q$  表示 输入向量  $\mathbf{X}$  对应的隶属函数值为  $\mu_A^1(\mathbf{X}) \sim \mu_A^q(\mathbf{X})$ 。模糊化方块将任一确定向量  $\mathbf{X}$  转换为相应的  $q$  个隶属函数值。输出向量空间用  $r$  个模糊数  $B^1 \sim B^r$  表示 任一输出向量  $\mathbf{Y}$  对应的隶属函数值为  $\mu_B^1(\mathbf{Y}) \sim \mu_B^r(\mathbf{Y})$ 。系统的  $r$  条模糊推理规则  $R^l$ ,  $l = 1 \sim r$ , 可以表示如下:

规则  $R^l$ : IF  $\mathbf{X}$  is  $A^{i_1(l)}$  OR  $\mathbf{X}$  is  $A^{i_2(l)}$  OR... $\mathbf{X}$  is  $A^{i_p(l)}$  THEN  $\mathbf{Y}$  is  $B^l$ ,  $l = 1 \sim r$  (5-2)

其中  $i_1(l), i_2(l), \dots, i_p(l) \in \{1 \sim q\}$  它们是第  $l$  条规则的前提所涉及输入空间模糊数的标号, 其个数为  $p, 1 \leq p \leq q$ 。对于某个  $\mathbf{X}$  第  $l$  条规则成立的可能性, 也就是  $\mathbf{X}$  对该条规则的隶属度  $\mu^l(\mathbf{X})$  可用下式计算:

$$\mu^l(\mathbf{X}) = \max\{\mu_A^{i_1(l)}(\mathbf{X}), \mu_A^{i_2(l)}(\mathbf{X}), \dots, \mu_A^{i_p(l)}(\mathbf{X})\} \quad (5-3)$$

第  $l$  条规则的结论是: 某个特定的  $\mathbf{Y}$  成为系统输出的可能性 (即隶属度) 为  $\mu_B^l(\mathbf{Y})$ 。系统中的模糊推理方块, 针对任何输入  $\mathbf{X}$  计算出  $\mu^l(\mathbf{X}), l = 1 \sim r$ 。去模糊化方块根据各  $\mu^l(\mathbf{X})$  和已存于系统中的各  $\mu_B^l(\mathbf{Y})$  求得系统输出的确定向量  $\mathbf{Y}$ 。

模糊逻辑推理系统的设计归结为两部分。第一, 结构学习: 即确定输入和输出空间模糊数的个数及其初始隶属函数; 确定规则条数及每条规则前提部分涉及的输入空间模糊数去模糊化算法。第二 参数学习 细调各参数。

由于应用领域的不同, 采用的模糊化、模糊推理和去模糊化方法不同, 所取的神经网络类型不同以及学习算法不同, 可以构成种类极为繁多的 FNN 来实现上述的模糊逻辑推理系统。这一节仅从应用领域出发来进行讨论。

### 5.2.2 用于函数逼近的 FNN

为了简化起见 以输出为 1 维向量 ( $M = 1$ ) 的情况为例, 来讨论函数逼近的问题。这时图 7-2 所示系统的输入至输出映射关系可以表示成  $y = f(\mathbf{X})$ , 系统待逼近的函数是  $y = \hat{f}(\mathbf{X})$ 。函数逼近是一个非常普遍的问题, 系统辨识和控制、时间序列预测等许多具体应用中都会遇到这个问题。

作为一个函数逼近器, 对 FNN 的要求是在系统的逼近误差 (一般以  $f(\mathbf{X})$  和  $\hat{f}(\mathbf{X})$  之间的均方误差作为衡量标准) 一定的条件下, 系统的复杂性尽量低 (以规则数的多少来衡量); 学习速度尽量快。此外, 当所提供的学习样本中存在一些偏离正确值很远的例外点 (outlier) 时, 不会使函数逼近效果产生较大劣化。

在各种实现函数逼近的 FNN 可能方案中 以 TSK (Takagi-Sugeno-Kang) 模型为基础的



方案独具优势。下面以输出为一维变量  $y$  的情况为例 介绍 TSK 模型。此模型的  $r$  条模糊推理规则表示为

$$\text{规则 } R^l: \mu^l(\mathbf{X}) \longrightarrow \mu_B^l(y) \quad l = 1 \sim r \quad (5-4)$$

对于任何  $\mathbf{X}$ ,  $\mu^l(\mathbf{X})$  为  $\mathbf{X}$  隶属于第  $l$  条规则前提的隶属度, 也就是针对此  $\mathbf{X}$  而言 第  $l$  条规则的可信度。 $\mu_B^l(y)$  如果是一个中心为  $b_0^l$ 、宽度为  $\sigma_0^l$  的对称隶属函数 (可以是梯形、三角形、高斯形或钟形), 那么可以通过下列去模糊化计算求得输出确定值:

$$y = \frac{\sum_{l=1}^r b_0^l \mu^l(\mathbf{X})}{\sum_{l=1}^r \mu^l(\mathbf{X})} \quad (5-5)$$

按此公式计算的模糊推理系统称为 0 阶 TSK 模型。注意  $\mu_B^l(y)$  的中心是  $b_0^l$  意味着  $\mu_B^l(b_0^l) = 1$ 。式( 5-5) 给出的是各条规则结论隶属函数中心值按其可信度进行加权取平均所得结果。如果计入结论隶属函数的宽度, 则此式修正为

$$y = \frac{\sum_{l=1}^r b_0^l \sigma_0^l \mu^l(\mathbf{X})}{\sum_{l=1}^r \sigma_0^l \mu^l(\mathbf{X})}$$

另一种修正计算公式是

$$y = \sum_{l=1}^r b_0^l \mu^l(\mathbf{X}) \quad (5-7)$$

1 阶 TSK 模型按下式进行去模糊化计算:

$$y = \frac{\sum_{l=1}^r (b_0^l + b_1^l x_1 + \cdots + b_N^l x_N) \mu^l(\mathbf{X})}{\sum_{l=1}^r \mu^l(\mathbf{X})} \quad (5-8)$$

此模型的优点是在给定的误差条件下用少量的规则实现非常复杂的映射。模型中待定的结构和参数是: 规则数  $r$ 、输入各模糊数的隶属函数  $\mu_A^l(\mathbf{X})$  和结论部分参数  $(b_0^l, b_1^l, \cdots, b_N^l)$ 。5.3 节将讨论有关的各种学习算法。TSK 模型的主要参考资料见文献[1]、文献[2]。

多种 FNN 已被证明是一种普适的函数逼近器 (universal function approximator) 即对于任何有限支连续单值非线性函数, 只要 FNN 的规则数足够多, 总能使逼近误差达到任意小值。有关的证明可以参见文献[3]、文献[4]。

### 5.2.3 用于分类的 FNN

设输入向量  $\mathbf{X}$  属于  $M$  种不同的类别  $C_m, m = 1 \sim M$ 。在用图 5-2 所示模糊逻辑推理系统实现此分类器时, 系统的  $M$  个输出  $y_1 \sim y_M$  与  $M$  个类别对应。当  $\mathbf{X} \in C_m$  时, 要求  $y_m = 1$  其他  $y_j = 0, j \neq m$ 。在系统实现上述分类要求时 模糊化方块仍产生  $q$  个与  $\mathbf{X}$  相应的隶属函数值  $\mu_A^1(\mathbf{X}) \sim \mu_A^q(\mathbf{X})$ ; 模糊推理方块则产生  $M$  个模糊推理结论  $\mu_B^1(y_1), \mu_B^2(y_2), \cdots, \mu_B^M(y_M)$  它们分别代表  $\mathbf{X}$  属于  $C_1, C_2, \cdots, C_M$  的隶属度。这样, 系统的规则数  $r$  即等于  $M$ ,  $M$  条推理规则可以表示为

$$\text{规则 } R^l: \mu^l(\mathbf{X}) \longrightarrow \mu_B^l(y_l), \quad l = 1 \sim M \quad (5-9)$$

其中前提仍可按式(5-3)计算,而结论与前提之间的关系可以直接写成

$$\mu_B^l(y_l) = \mu^l(\mathbf{X}), \quad l = 1 \sim M \quad (5-10)$$

去模糊化方块比较各  $\mu_B^l(y_l)$  取其最大者,令相应输出为 1 其他输出为 0 即

$$y_m = \begin{cases} 1, & \mu_B^m(y_m) = \max_i \{\mu_B^i(y_i)\} \\ 0, & \text{其他} \end{cases} \quad (5-11)$$

对 FNN 分类器的要求是,分类错误率不超过给定的指标以及系统结构尽量简单、学习速度尽量快等。系统的结构学习可确定输入空间模糊数的个数  $q$  初步确定每个模糊数的隶属函数  $\mu_A^i(\mathbf{X})$  确定  $M$  条模糊推理规则的前提  $\mu^l(\mathbf{X})$  (即对式 5-3) 中的  $\mu_A^{i(l)}(\mathbf{X})$  等进行选择)。系统的参数学习则对已初步确定的各  $\mu_A^i(\mathbf{X})$  中所用的参数进行细调。无论在结构学习阶段还是参数学习阶段,都需要用到由  $P$  个学习样本  $\{\mathbf{X}_p, d_p\}, p = 1 \sim P$  构成的训练集 其中  $d_p$  是  $\mathbf{X}_p$  所属的理想类别的标号。

在若干应用中,要求系统作出软判决,即提供某个  $\mathbf{X}$  属于所有  $M$  个类别的可能值(即隶属度值)。这时可取消去模糊方块,直接输出各  $\mu_B^l(y_l)$  即可。

#### 5.2.4 用于聚类的 FNN

设有  $P$  个  $N$  维向量  $\mathbf{X}_p, p = 1 \sim P$  构成的训练集。聚类的目的是将它们分成  $M$  个聚类区  $CL_i, i = 1 \sim M$ 。设其“质心”为  $\mathbf{W}_i, i = 1 \sim M$  每个  $\mathbf{W}_i$  也是  $N$  维向量。那么对于任  $\mathbf{X}_p$  若满足条件:  $\|\mathbf{X}_p - \mathbf{W}_k\| \leq \|\mathbf{X}_p - \mathbf{W}_i\|, i \neq k$  则  $\mathbf{X}_p$  归属于第  $k$  个聚类区  $CL_k$ 。其中  $\|\cdot\|$  表示欧氏距离。在一般的聚类学习算法中,求出  $M$  个质心  $\mathbf{W}_i$  使得下列目标函数  $J$  达到最小:

$$J = \sum_{i=1}^M \sum_{p=1, \mathbf{X}_p \in CL_i}^P d_{pi}^2 \quad (5-12)$$

其中  $d_{pi} = \|\mathbf{X}_p - \mathbf{W}_i\|$  是  $\mathbf{X}_p$  与第  $i$  聚类区  $CL_i$  质心之间的欧氏距离  $\mathbf{X}_p \in CL_i$  意味着  $d_{pi} < d_{pj}, j = 1 \sim M, j \neq i$ 。

按照 FCM(fuzzy C-Means) 准则的模糊聚类算法则将目标函数修正为下列  $J_F$ :

$$J_F = \sum_{i=1}^M \sum_{p=1}^P \mu_{pi}^m d_{pi}^2 \quad (5-13)$$

其中  $d_{pi}$  的定义与  $J$  的情况相同,  $\mu_{pi}$  是  $\mathbf{X}_p$  对  $CL_i$  的隶属度 当  $\mathbf{X}_p = \mathbf{W}_i$  时  $\mu_{pi} = 1$  当  $\mathbf{X}_p$  与  $\mathbf{W}_i$  的距离变远时  $\mu_{pi}$  逐渐下降。 $m$  是一个起调节作用的非负系数(一般取  $m \geq 1$ )。可以看到 若  $\mathbf{X}_p \in CL_i$  时  $\mu_{pi} = 1$  若  $\mathbf{X}_p \notin CL_i$  时  $\mu_{pi} = 0$  则  $J_F$  退化为  $J$ 。由于模糊聚类算法将各个聚类区边界间的突变通过模糊化转换为渐变,使得形成的聚类区划分更加合理。

从实现角度看,模糊聚类系统与模糊分类系统十分相似,由模糊化、模糊推理和去模糊化三部分组成且每部分的工作原理也类似。但是,分类器的  $M$  个输出表示由外界规定的某个输入  $\mathbf{X}$  所属的类别标号;而聚类器的  $M$  个输出是反映  $\mathbf{X}$  所属输入空间拓扑特性的  $M$  个聚类区标号。前者的学习过程中必然包括有监督学习的阶段,而后者全部学习过程都是自组织的。当然,二者给出的  $M$  个输出可以是硬判决的结果,即  $\mathbf{X}$  的类别标号或聚

类区标号，也可以是软判决的结果，即  $\mathbf{X}$  对于各类别或聚类区的隶属度值；模糊聚类系统中一般取后一方式。

模糊聚类系统将输入向量空间划分为  $M$  个模糊聚类区，在工作时对于每个输入  $\mathbf{X}$  给出其对各模糊聚类区的隶属度值。这除了构成一种数据压缩方案以外，还可以用于作为 FNN 函数逼近器或 FNN 分类器中的模糊化部分。

有关采取 FCM 准则的模糊聚类基本原理可以参阅文献[5]～[7]。

## 5.3 实现函数映射的 FNN

### 5.3.1 网络结构

此 FNN 是一个 5 层前向网络，其中第 1 层有  $N$  个神经元，其第  $i$  神经元的输入和输出皆等于输入向量  $\mathbf{X}$  的第  $i$  分量  $x_i$ 。如将此输入层除外，则网络是一个 4 层前向网络。为了简化起见，下面只讨论如何用它来实现一个一阶 TSK 模型下的模糊逻辑推理系统（0 阶 TSK 模型可以看作一个特例）。系统的运行算法由式 (5-2)、(5-3) 和 (5-8) 给出。

首先，讨论式 (5-2) 中规则  $R^i$  的前提部分。其中任一项“ $\mathbf{X}$  is  $A^i$ ”表示  $\mathbf{X}$  对  $A^i$  的隶属度（ $i$  为  $i_1(l)$  或  $i_2(l)\cdots$ ）这归结为隶属度函数  $\mu_A^i(\mathbf{X})$  的计算。通过下列逻辑运算公式，这个  $N$  维隶属函数的计算可以归结为  $N$  个 1 维隶属函数的运算，

$$\mathbf{X} \text{ is } A^i = x_1 \text{ is } A_1^i \text{ and } x_2 \text{ is } A_2^i \text{ and } \cdots x_N \text{ is } A_N^i \quad (5-14)$$

其中  $A_1^i \cdots A_N^i$  是  $N$  个 1 维模糊数。假设  $x_1 \sim x_N$  对它们的隶属函数是  $\mu_{A_1^i}(x_1) \sim \mu_{A_N^i}(x_N)$ ，那么可以由这些 1 维隶属函数计算出  $\mu_A^i(\mathbf{X})$ 。在模糊逻辑中，按照“and”运算，常取下列几种计算方法。

第一，按照取最小的方式，

$$\mu_A^i(\mathbf{X}) = \min_{n=1 \sim N} \{ \mu_{A_1^i}(x_1), \cdots, \mu_{A_N^i}(x_N) \} \quad (5-15)$$

第二，按照相乘方式，

$$\mu_A^i(\mathbf{X}) = \mu_{A_1^i}(x_1) \cdot \mu_{A_2^i}(x_2) \cdot \cdots \cdot \mu_{A_N^i}(x_N) \quad (5-16)$$

第三，按照相加方式，

$$\mu_A^i(\mathbf{X}) = \frac{1}{N} [\mu_{A_1^i}(x_1) + \mu_{A_2^i}(x_2) + \cdots + \mu_{A_N^i}(x_N)] \quad (5-17)$$

研究结果表明，相乘方式的性能优于取最小方式（见文献[8]）且计算更为方便，所以基于 TSK 模型的 FNN 中采用相乘方式的比较多。相加方式在 Fuzzy ART 型 FNN 中用得比较多。

下面讨论  $\mu_{A_n^i}(x_n)$  的计算问题，首先须要确定这些函数所取的类型。图 5-3 所示是几种常用的隶属函数类型，分别讨论如下。

(1) 梯形

一个对称梯形隶属函数  $\mu_{A_n^i}(x_n)$  的示例如图 5-3(a) 示，它的描述参数是中心  $M_n^i$ 、上顶宽  $h_n^i$  和下底宽  $g_n^i$ 。它的另一组描述参数是上顶的左端点（或称 MIN 点） $U_n^i$ 、右端点（或称 MAX 点） $N_n^i$  以及衰减系数  $\gamma_n^i$ （ $\gamma_n^i$  越大则边缘过渡区衰减得越快）。当用后者时，隶属函

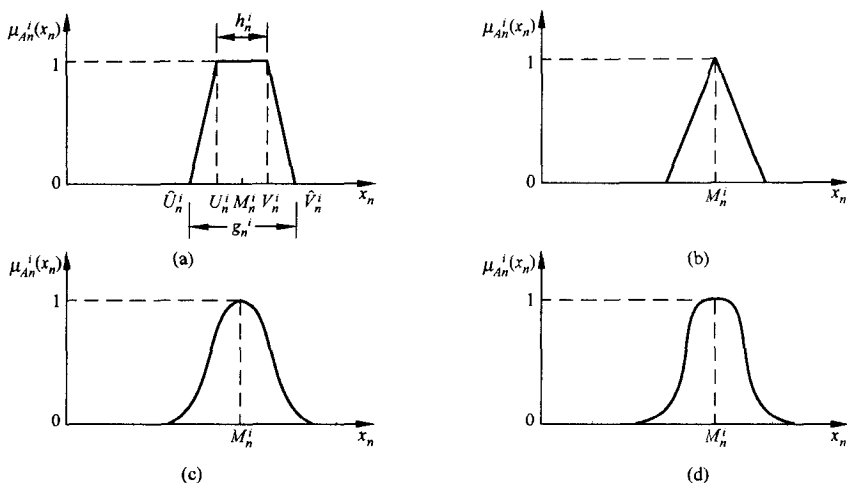


图 5-3 几种常用的隶属函数

数可用下式计算（注意： $\gamma_n^i > 0$ ）：

$$\mu_{An}^i(x_n) = 1 - \varphi(x_n - V_n^i, \gamma_n^i) - \varphi(U_n^i - x_n, \gamma_n^i)$$

$$\varphi(x_n, \gamma_n^i) = \begin{cases} 1, & x_n \gamma_n^i > 1 \\ x_n \gamma_n^i, & 0 \leq x_n \gamma_n^i \leq 1 \\ 0, & x_n \gamma_n^i < 0 \end{cases} \quad (5-18)$$

如果  $\mu_{An}^i(x_n)$  是非对称的梯形函数，则需要用 4 个参数来描述，即除了上顶的左右端点  $U_n^i$  和  $V_n^i$  以外，还包含下底的左右端点  $\hat{U}_n^i$  和  $\hat{V}_n^i$ 。

#### (2) 三角形

这是梯形的一个特例（ $U_n^i, M_n^i, V_n^i$  并为一个点）如图 5-3(b) 所示。

#### (3) 钟形

其计算公式如下：

$$\mu_{An}^i(x_n) = \frac{1}{1 + \left[ \left( \frac{x_n - M_n^i}{\sigma_n^i} \right)^2 \right]^{\gamma_n^i}} \quad (5-19)$$

其中的 3 个参数是中心  $M_n^i$ 、宽度  $\sigma_n^i$  ( $\sigma_n^i > 0$ ) 和衰减速度  $\gamma_n^i$  ( $\gamma_n^i > 0$ )。其形状见图 5-3(c)。

#### (4) 高斯形

其计算公式为

$$\mu_{An}^i(x_n) = \exp \left\{ - \left[ \left( \frac{x_n - M_n^i}{\sigma_n^i} \right)^2 \right]^{\gamma_n^i} \right\} \quad (5-20)$$

其中  $M_n^i, \sigma_n^i, \gamma_n^i$  3 个参数的意义与钟形情况相同。其形状见图 5-3(d)。

基于 TSK 模型的 FNN 中，梯形和高斯形的隶属函数用得比较多。

图 5-4 所示是一个基于一阶 TSK 模型的完整 5 层 FNN 的结构图。为简化起见 系统的输出是 1 维变量  $y_6$  系统 2 至 5 层的功能分叙如下。

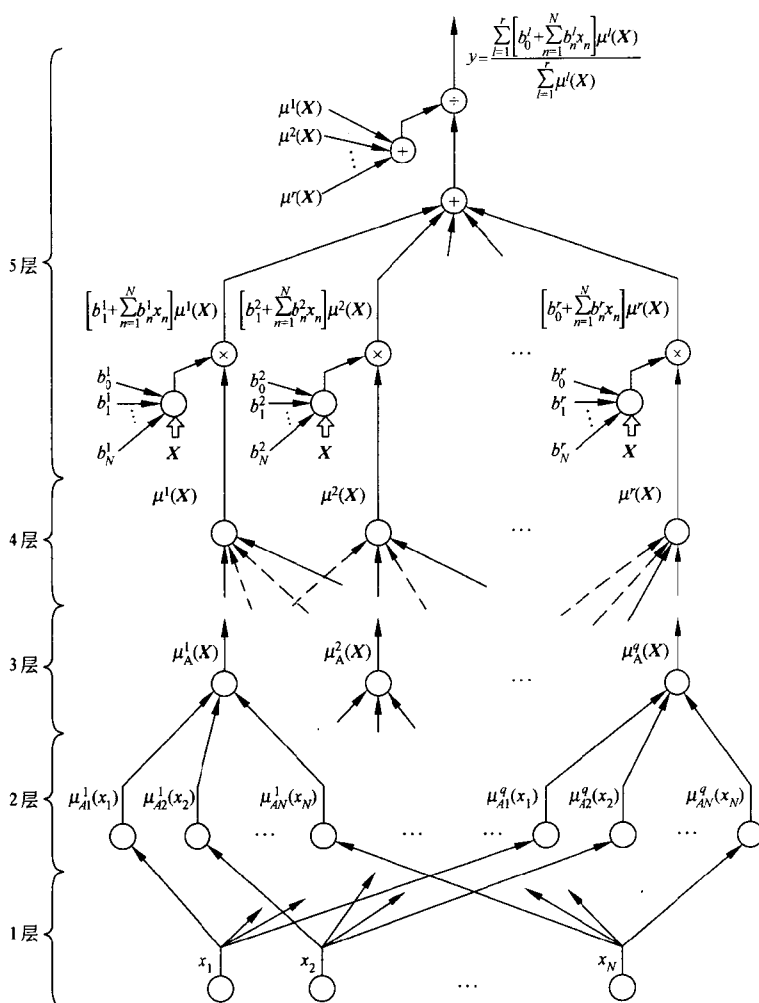


图 5-4 基于一阶 TSK 的 FNN 结构图

(1) 第 2 层

共含  $q \times N$  个神经元 每  $N$  个神经元构成一组，共  $q$  组。每组神经元完成 1 维隶属函数  $\mu_{A1}^i(x_1) \cdots \mu_{AN}^i(x_N)$  的计算  $i = 1 \sim q$ 。

(2) 第 3 层

共含  $q$  个神经元，每个神经元完成一个  $N$  维隶属函数  $\mu_A^i(\mathbf{X})$  的计算  $i = 1 \sim q$ 。计算可以在式 (5-15) ~ (5-17) 中择取一种进行。当采取式 (5-16) 且 1 维隶属函数取高斯形时  $\mu_A^i(\mathbf{X})$  可表示为

$$\mu_A^i(\mathbf{X}) = \prod_{n=1}^N \exp \left\{ - \left[ \left( \frac{x_n - M_n^i}{\sigma_n^i} \right)^2 \right]^{r_n^i} \right\}, \quad i = 1 \sim q \quad (5-21)$$

(3) 第 4 层

共包含  $r$  个神经元，每个神经元完成式 (5-2) 所描述的推理运算，即执行模糊规则  $R^l$ ， $l = 1 \sim r$ 。其形式化的表示由式 (5-3) 给出，其前提部分取自  $\mu_A^i(\mathbf{X})$ ， $i = 1 \sim q$  (在图 5-4

中用实线和虚线示意地表示某条规中被选中和未选中的  $\mu'_A(\mathbf{X})$ 。其结论即为输出  $\mu'(\mathbf{X})$ 。

#### (4) 第 5 层

完成去模糊化运算。在这一层利用第 4 层提供的  $\mu^1(\mathbf{X}) \sim \mu^r(\mathbf{X})$ 、输入向量  $\mathbf{X}$  和存储的  $r$  组系数  $b'_0, b'_1, \dots, b'_N, l=1 \sim r$  根据式 (5-8) 计算得输出  $y_0$ 。这可表示为  $y = f(\mathbf{X})$ 。

网络中待定的参数是： $q, r, \mu'_A(\mathbf{X})$  (分解成  $N$  个  $\mu'_{An}(x_n)$  相乘) 推理规则 ( $\mu'(\mathbf{X})$  的前提是哪些  $\mu'_A(\mathbf{X})$ ) 以及  $r$  组系数  $(b'_0, b'_1, \dots, b'_N), l=1 \sim r$ 。目标是通过适当的学习后网络得以用尽可能简单的结构 ( $q$  和  $r$  尽量小) 实现  $y = f(\mathbf{X})$  对某个待逼近函数  $y = \hat{f}(\mathbf{X})$  的逼近误差达到预定的指标。结构学习阶段确定  $q, r, \mu'_A(\mathbf{X}), i=1 \sim q$  (只粗略地确定) 和推理规则。参数学习阶段对组成  $\mu'_A(\mathbf{X})$  的  $\mu'_{An}(x_n)$  中的参数  $M_n^i, \sigma_n^i, \gamma_n^i$  以及各  $b'_j$  进行精细调节。

### 5.3.2 结构学习之一 —— 输入空间的模糊划分

结构学习的第一项内容是将输入向量  $\mathbf{X}$  的空间划分为  $q$  个模糊区。这里的问题是确定  $q$  值和每个区的隶属函数  $\mu'_A(\mathbf{X}), i=1 \sim q$ 。

一种最简单的方法是将  $\mathbf{X}$  的每一维分成若干个等间隔区，每个区有相同的隶属函数。早期在用手工作法设计模糊推理系统时，即用此法。图 5-5 所示是  $\mathbf{X}$  为 2 维向量情况下，每一维变量 ( $x_1$  和  $x_2$ ) 都分成 3 个区的示例；从隶属函数看出，各区有一定的交叠且靠近边缘区的宽度是中部区宽度的一半 (假设  $\mathbf{X}$  的变化范围在图示的方框内) 这时输入空间划分成 9 个模糊区 即  $q=9$  每个区的隶属函数  $\mu'_A(\mathbf{X})$  可由两个维的 1 维隶属函数相乘得到。

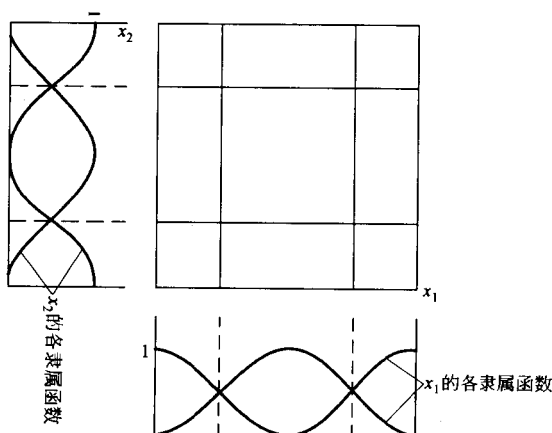


图 5-5  $\mathbf{X}$  为 2 维时的隶属函数

这种简单方法只能适用于非常简单的情况，即  $\mathbf{X}$  的维数很低的情况。若  $\mathbf{X}$  的维数  $N$  较大，把每一维向量分成  $K$  个区，那么  $q = N^K$  将随着  $N$  和  $K$  的增大而增至非常大的数值。此外，这种等间隔分区法也完全没有考虑  $\mathbf{X}$  在其空间中的分布疏密状况。下面将介绍若干种有效的划分方案。假设训练集是  $(\mathbf{X}_p, \hat{y}_p), p=1 \sim P$  它们分别表示  $P$  对训练样本

的输入向量和理想输出。在进行输入空间划分时只用到  $\mathbf{X}_p, p = 1 \sim P$ 。

### 1. 基于 Fuzzy ART 的空间划分算法

将在 5.4 节中讨论。

### 2. 基于 FCM 的空间划分算法

将在 5.5 节中讨论。

### 3. 基于遗传算法 (genetic algorithm, GA) 的空间划分算法

见文献[9]、[10] 并参考第 6 章。

### 4. 相似度比较法<sup>[11]</sup>

设各前提隶属函数  $\mu_A^i(\mathbf{X})$  及各维隶属函数  $\mu_{A_n}^i(x_n)$  分别用式 (5-16) 和式 (5-20) 计算, 且设  $\gamma_n^i = 1$  这样可产生式 (5-21)。学习时按时间节拍  $t = 1, 2, \dots$  从训练集中依次取出输入向量  $\mathbf{X}$  并将其表示为  $\mathbf{X}(t)$ 。学习过程如下。

(1) 设置若干参数和函数: 初始 1 维隶属函数宽度  $\sigma_0$ ; 相似度比较门限函数  $\mu_T(t)$  令  $0 < \mu_T(t) < 1$  当  $t = 1$  时  $\mu_T(t)$  接近于 1 当  $t$  增加时,  $\mu_T(t)$  逐渐降至一适当值; 一维隶属函数归并门限  $\rho_T(t)$  其值在  $t = 1$  时略小于 1 随  $t$  增加而略下降;  $t$  时刻有  $q(t)$  个前提模糊数。

(2) 当  $t = 1$  输入  $\mathbf{X}(1) = [x_1(1), \dots, x_N(1)]$  令  $q(1) = 1$ 。令

$$\mu_A^1(\mathbf{X}) = \prod_{n=1}^N \exp \left[ - \left( \frac{x_n - M_n^1}{\sigma_n^1} \right)^2 \right] \quad (5-22)$$

其中  $M_n^1 = x_n(1), \sigma_n^1 = \sigma_0$ 。

(3) 当  $t > 1$  输入  $\mathbf{X}(t) = [x_1(t), \dots, x_N(t)]$ 。这时已有  $q(t-1)$  个隶属函数  $\mu_A^i(\mathbf{X})$ ,  $i = 1 \sim q(t-1)$ 。设

$$J(t) = \operatorname{argmax}_{i=1 \sim q(t-1)} [\mu_A^i(\mathbf{X}(t))] \quad (5-23)$$

若  $\mu_A^{J(t)}(\mathbf{X}(t)) \geq \mu_T(t)$  则不作任何变动 转入  $t+1$  步运算。且令  $q(t) = q(t-1)$ 。

若  $\mu_A^{J(t)}(\mathbf{X}(t)) < \mu_T(t)$  则增加一个模糊数 这时令  $q(t) = q(t-1) + 1$ 。新增加模糊数具有下列隶属函数:

$$\mu_A^{q(t)}(\mathbf{X}) = \prod_{n=1}^N \exp \left[ \left( \frac{x_n - M_n^{q(t)}}{\sigma_n^{q(t)}} \right)^2 \right] \quad (5-24)$$

其中

$$M_n^{q(t)} = x_n(t) \quad \sigma_n^{q(t)} = -\beta \ln [\mu_A^{J(t)}(\mathbf{X}(t))]$$

$\beta$  是一个预置的固定常数。其他前  $q(t-1)$  个隶属函数不作任何更改。显然, 当  $J_T(t)$  设置得越接近于 1 时划分的区间数越多, 即空间划分得越细。

(4) 1 维隶属函数的合并。每一个  $\mu_A^i(\mathbf{X})$  由  $N$  个 1 维隶属函数构成, 如不采取措施, 随着  $q(t)$  的增加, 网络中需要计算的 1 维隶属函数个数将线性增加 (即为  $Nq(t)$ )。其实 同一维的各个 1 维隶属函数中有不少是很接近的, 如予以合并则可使网络简化, 从而减少网络运行时的计算量。为了比较同一维的两个 1 维隶属函数  $\mu_{A_n}^\alpha(x_n)$  和  $\mu_{A_n}^\beta(x_n)$  的相似度, 以决定其能否合并, 定义其相似度  $E(\alpha_n, \beta_n)$  如下:

$$E(\alpha_n, \beta_n) = \frac{|\alpha_n \cap \beta_n|}{|\alpha_n \cup \beta_n|} \quad (5-25)$$

$\alpha_n \cap \beta_n$  和  $\alpha_n \cup \beta_n$  分别表示模糊数  $\alpha_n$  和  $\beta_n$  的“交”和“联”。 $|\alpha_n \cap \beta_n|$  和  $|\alpha_n \cup \beta_n|$  可用下式计算：

$$|\alpha_n \cap \beta_n| = \frac{1}{2} \frac{h^2(M_n^\beta - M_n^\alpha + \sqrt{\pi}(\sigma_n^\alpha + \sigma_n^\beta))}{\sqrt{\pi}(\sigma_n^\alpha + \sigma_n^\beta)} + \frac{1}{2} \frac{h^2(M_n^\beta - M_n^\alpha + \sqrt{\pi}(\sigma_n^\alpha - \sigma_n^\beta))}{\sqrt{\pi}(\sigma_n^\beta - \sigma_n^\alpha)} + \frac{1}{2} \frac{h^2(M_n^\beta - M_n^\alpha - \sqrt{\pi}(\sigma_n^\alpha + \sigma_n^\beta))}{\sqrt{\pi}(\sigma_n^\alpha - \sigma_n^\beta)} \quad (5-26)$$

其中  $M_n^\alpha, M_n^\beta, \sigma_n^\alpha, \sigma_n^\beta$  分别表示  $\mu_n^\alpha(x_n)$  和  $\mu_n^\beta(x_n)$  的中心和宽度参数且设  $M_n^\alpha \geq M_n^\beta$  函数  $h(u)$  定义为

$$h(u) = \max(0, u) \quad (5-27)$$

$$|\alpha_n \cup \beta_n| = \sqrt{\pi} \sigma_n^\alpha + \sqrt{\pi} \sigma_n^\beta - |\alpha_n \cap \beta_n| \quad (5-28)$$

当  $\sigma_n^\alpha = \sigma_n^\beta$  且  $M_n^\alpha = M_n^\beta$  时  $E(\alpha_n, \beta_n) = 1$ 。下面给出归并算法。设  $t$  时刻下第  $n$  维变量含  $k_n(t)$  个隶属函数且  $k_n(1) = 1, n = 1 \sim N$ 。当  $t > 1$  时，令新建的  $\mu_{\lambda_n}^{q(t)}(x_n)$  与已建的  $k_n(t-1)$  个 1 维隶属函数  $\mu_{\lambda_n}^p(x_n), p = 1 \sim k_n(t-1)$  进行比较，若满足下列条件：

$$E_{n, \max}(t) = \max_{p=1 \sim k_n(t-1)} \{E[\mu_{\lambda_n}^{q(t)}(x_n), \mu_{\lambda_n}^p(x_n)]\} \geq \rho_T(t) \quad (5-29)$$

则无需在第  $n$  维建立新的 1 维隶属函数。这时令  $k_n(t) = k_n(t-1)$  且使用已建的 1 维隶属函数中与  $\mu_{\lambda_n}^{q(t)}(x_n)$  最相似者代替之。反之若此条件不满足则应令  $k_n(t) = k_n(t-1) + 1$  且将  $\mu_{\lambda_n}^{q(t)}(x_n)$  作为新增加的  $\mu_{\lambda_n}^{k_n(t)}(x_n)$

##### 5. MMC(modified mountain clustering) 法<sup>[12,13,14]</sup>

已知训练集为  $X_p, p = 1 \sim P$  其中  $X_p$  为归一化  $N$  维向量，即下列条件得到满足：

$\|X_p\|_{\max} = \max_{p=1 \sim P} \{\|X_p\|\} = 1$ 。若此条件不满足只需令诸  $X_p$  除以  $\|X_p\|_{\max}$  即可得所需的归一化向量。MMC 算法如下列。

(1) 计算  $X_1 \sim X_P$  的“位”(“potential”)  $\Psi_p^{(1)}$  如下：

$$\Psi_p^{(1)} = \sum_{l=1}^P \exp\left(-\frac{4}{r_a^2} \|X_p - X_l\|^2\right), \quad p = 1 \sim P \quad (5-30)$$

$\Psi_p^{(1)}$  越大表明  $X_p$  附近聚集的训练向量样本越多。 $r_a$  是一个待定的邻域半径系数。

(2) 设  $\Psi_1^* = \max_{p=1 \sim P} \Psi_p^{(1)}$  并将与  $\Psi_1^*$  相应的训练向量用  $X_1^*$  表示。将  $X_1^*$  定为第一个模糊聚类区的中心。

(3) 将  $X_1^*$  除外后计算其他  $X_1 \sim X_P$  的位  $\Psi_p^{(2)}$  如下：

$$\Psi_p^{(2)} = \Psi_p^{(1)} - \Psi_1^* \exp\left(-\frac{4}{r_b^2} \|X_p - X_1^*\|^2\right), \quad p = 1 \sim P \quad \text{且} \quad X_p \neq X_1^* \quad (5-31)$$

$r_b$  是一个待定的邻域半径系数，较优的选择是  $r_b = 1.5r_a$ 。 $\Psi_p^{(2)}$  越大，表明  $X_p$  附近聚集的训练向量样本越多且  $X_p$  与  $X_1^*$  的距离足够远。

(4) 设  $\Psi_2^* = \max_{p=1 \sim P} (\Psi_p^{(2)})$  与  $\Psi_2^*$  相应的训练向量用  $X_2^*$  表示。将  $X_2^*$  定为第二个模糊聚类区的中心（注意，计算  $\Psi_2^*$  时已将  $X_1^*$  除外）

(5) 对于  $k \geq 2$  将  $X_1^*, X_2^*, \dots, X_k^*$  除外后计算  $X_1 \sim X_P$  的位  $\Psi_p^{(k+1)}$ ：



$$\Psi_p^{(k+1)} = \Psi_p^{(k)} - \Psi_k^* \exp\left(-\frac{4}{r_b^2} \|\mathbf{X}_p - \mathbf{X}_k^*\|^2\right), \quad p = 1 \sim P \quad \text{且} \quad \mathbf{X}_p \neq \mathbf{X}_1^*, \dots, \mathbf{X}_k^* \quad (5-32)$$

$$\text{设} \quad \Psi_{k+1}^* = \max_{p=1 \sim P} (\Psi_p^{(k+1)}) \quad \text{其中} \quad \mathbf{X}_p \neq \mathbf{X}_1^*, \dots, \mathbf{X}_k^* \quad (5-33)$$

(6) 若  $\Psi_{k+1}^* > \bar{\epsilon} \Psi_1^*$  则将与  $\Psi_{k+1}^*$  相应的向量  $\mathbf{X}_p$  用  $\mathbf{X}_{k+1}^*$  表示。  $\mathbf{X}_{k+1}^*$  成为第  $k+1$  个聚类区的中心。然后令  $k$  增加 1 再转回 ⑤ 项计算。  $\bar{\epsilon}$  是一个可调节的系数，文献 [11] 中定  $\bar{\epsilon} = 0.5$ 。

(7) 若  $\Psi_{k+1}^* < \epsilon \Psi_1^*$  则停止运算。将此刻的  $k$  值定为聚类区的个数  $q$  且  $\mathbf{X}_1^*, \dots, \mathbf{X}_k^*$  是它们的中心。  $\epsilon$  是一个可调系数，文献 [11] 中定  $\epsilon = 0.15$ 。

(8) 若  $\epsilon \Psi_1^* \leq \Psi_{k+1}^* \leq \bar{\epsilon} \Psi_1^*$ ，则检查下列条件是否满足：

$$\frac{d_{\min}}{r_a} + \frac{\Psi_{k+1}^*}{\Psi_1^*} \geq 1 \quad (5-34)$$

$$\text{其中} \quad d_{\min} = \min_{l=1 \sim k} (\|\mathbf{X}_{k+1}^* - \mathbf{X}_l^*\|) \quad (5-35)$$

若满足，则接受  $\mathbf{X}_{k+1}^*$  成为第  $k+1$  个聚类中心。再令  $k$  增加 1 转回 (5) 项计算。若不满足，则废弃诸  $\Psi_p^{(k+1)}$  之最大者，而选次大者作为  $\mathbf{x}_{k+1}$ ，这可以表示为

$$\Psi_{k+1}^* = 2nd \max_{p=1 \sim P} (\Psi_p^{(k+1)}) \quad \text{其中} \quad \mathbf{X}_p \neq \mathbf{X}_1^*, \dots, \mathbf{X}_k^* \quad (5-36)$$

然后再进行 (5-34) 式所规定条件的检验并决定是否接受与新的  $\Psi_{k+1}^*$  相应的  $\mathbf{X}_p$  作为第  $k+1$  聚类中心  $\mathbf{X}_{k+1}^*$ 。这一过程可继续下去。若能可接受者则令  $k$  增加 1 转回 (5) 项计算。若始终不可接受，则停止运算。将聚类区个数  $q$  定为此刻的  $k$  值。其中心为  $\mathbf{X}_1^*, \dots, \mathbf{X}_q^*, q = k$ 。

需说明 参数  $r_a$  的选择及  $\bar{\epsilon}, \epsilon$  的选择将决定聚类区的个数  $q$ 。  $r_a$  越小则  $q$  越大。文献 [11] 中定  $r_a = 0.35$ 。最后形成的输入空间的  $q$  个隶属函数  $\mu_A^i(\mathbf{X}), i = 1 \sim q$  由  $\mathbf{X}_1^*, \dots, \mathbf{X}_q^*$  决定，如下式所列：

$$\mu_A^i(\mathbf{X}) = \exp\left(-\frac{4}{r_a^2} \|\mathbf{X} - \mathbf{X}_i^*\|^2\right), \quad i = 1 \sim q \quad (5-37)$$

设  $\mathbf{X}_i^* = [x_{i1}^*, \dots, x_{iN}^*]$  则可将上式写成

$$\mu_A^i(\mathbf{X}) = \prod_{n=1}^N \mu_{A_n}^i(x_n) = \prod_{n=1}^N \exp\left[-\frac{4}{r_a^2} (x_n - x_n^*)^2\right] \quad (5-38)$$

$\mu_{A_n}^i(x_n)$  是第  $i$  聚类区第  $n$  维的隶属函数。

## 6. 切割法

前述 4、5 节所列的输入空间划分方法可以称为由一点至全体的增长式法。切割法则由整体开始，通过一系列切割，将输入空间不断划分，直至达到一个预定指标为止。切割时从  $\mathbf{X} = [x_1, \dots, x_n]$  选出一个恰当的维，例如第  $S$  维，在变量  $x_s$  中选出一恰当点将整个输入空间切成两部分，然后，对这两部分再选恰当维的恰当点进行切割。图 5-6 所示是对一个 2 维输入向量  $\mathbf{X} = [x_1, x_2]$  空间进行切割的示例，其中，  
.....表示第一刀，第二刀.....。问题在于如何选择适当的维以及如何选择适当的切割点。下面介绍的是诸多方案中的两种。

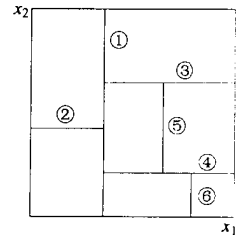


图 5-6 一个切割示例

(1) 按模糊目标函数进行切割的登山算法<sup>[15]</sup>

图 5-6 所示的切割图形称为模糊  $k-d$  树，其中每切一刀就将原来的一个空间分成了两个空间，假设切割后全部输入空间共含  $Q$  个模糊划分区，各用模糊隶属函数  $\mu_A^i(X)$ ， $i = 1 \sim Q$  表示（各相邻隶属函数有一定交叠），那么对于训练集  $X_p, p = 1 \sim P$  可以建立下列模糊目标函数：

$$J = J_D + J_T = \sum_{j=1}^P \sum_{k=1}^P \left\{ \left[ \sum_{i=1}^Q (\mu_j^i - \mu_k^i)^2 \right] - d_{jk}^2 \right\}^2 + \sum_{k=1}^P \sum_{i=1}^Q (\mu_k^i)^2 d_{ik}^2 \quad (5-39)$$

其中  $\mu_j^i = \mu_A^i(X_j)$ ， $\mu_k^i = \mu_A^i(X_k)$ ， $d_{jk}^2 = \|X_j - X_k\|^2$ ， $d_{ik}^2 = \|X_i^* - X_k\|^2$ ， $X_i^*$  是第  $i$  个模糊划分区的中心。 $J_D$  是一个衡量划分区内样本密集度的目标函数，如果同一区内各训练样本越密集（即越紧凑）则  $J_D$  越小，这种情况称为相应的隶属函数具有强支。 $J_T$  是衡量划分区的中心  $X_i^*$  是否具有典型性（即能代表区内诸训练样本）的量度， $J_T$  越小则典型性越高。实际上  $J_T$  就是前述 FCM 中的目标函数  $J_K$ （见式 5-13）只需在该式中取  $m = 2$ 。这样每一刀的切割维和切割点的选择应使其所导致的  $J$  达到最小。应注意，每切一刀只改变某维的 1 维区域划分和相应隶属函数的改变，而其他各维的隶属函数变动较小。

(2) 与输出区分能力相联系的切割法<sup>[16]</sup>

此法将同时应用训练集的输入和输出部分，即  $(X_p, \hat{y}_p)$ ， $p = 1 \sim P$  其中  $\hat{y}_p$  是与  $X_p$  对应的理想输出。此方法的基本思路是若将某一空间切割为两部分且每一部分用隶属函数  $\mu_A^{(1)}(X)$  和  $\mu_A^{(2)}(X)$  表示时，可用下列特征参数  $C_{bs}$ ， $s = 1, 2$ ，来表征这两部分映射的效果：

$$C_{bs} = \frac{\sum_{p=1}^P \hat{y}_p \mu_A^{(s)}(X_p)}{\sum_{p=1}^P \mu_A^{(s)}(X_p)}, \quad s = 1, 2 \quad (5-40)$$

若  $|C_{b1} - C_{b2}|$  越大，则切割后的两部分越具有互异映射效果。若  $|C_{b1} - C_{b2}| = 0$  说明二部分毫无差异。这样，所选的切割维和切割点应为使  $|C_{b1} - C_{b2}|$  为最大者。文献[16]建议，不必求每一维的最佳切割点（那样做时计算量太大），而只选择该维取值范围的 0.35, 0.5, 0.65 这 3 个点做切割并选其最佳者即可。

切割法的一个问题是每一刀都彻底地将一个空间切为两半，有时这样做并不有利，可见文献[15]的讨论。

## 7. 模糊曲线法<sup>[17]</sup>

这个方法是按输入向量  $X$  的每一维来考察其对输出  $y$  的影响，并在此基础上建立输入向量空间的模糊划分。仍设训练集为  $(X_p, \hat{y}_p)$ ， $p = 1 \sim P$ 。对于  $X$  的每一维分量  $x_n$  可以用下列公式计算出一条模糊曲线：

$$C_n(x_n) = \frac{\sum_{p=1}^P \varphi_n^p(x_n) \cdot \hat{y}_p}{\sum_{p=1}^P \varphi_n^p(x_n)}, \quad n = 1 \sim N \quad (5-41)$$

其中

$$\varphi_n^p(x_n) = \exp \left\{ - \left[ \frac{x_n - x_{pn}}{\sigma_n} \right]^2 \right\} \quad (5-42)$$

$x_{pn}$  是  $\mathbf{X}_p$  的第  $n$  维分量。若训练集中各  $x_{pn}$  的上、下界分别是  $\bar{x}_n$  和  $x_n$  则可令  $\sigma_n = 0.2(\bar{x}_n - x_n)$ 。

$N$  条模糊曲线  $C_n(x_n)$ ,  $n = 1 \sim N$  表征了每一维输入分量对输出  $y$  的影响。若  $C_n(x_n)$  越平坦 则  $x_n$  对输出的影响越小。文献[17] 中给了一个示例,  $N = 3, P = 20, x_1 = 6.9, \bar{x}_1 = 26.6, x_2 = 1.3, \bar{x}_2 = 3.8, x_3 = 4.5, \bar{x}_3 = 10.3, \hat{y} = 1.8, \bar{y} = 4.9$  ( $\hat{y}$  和  $\bar{y}$  是训练集中  $\hat{y}_p$  的最小值和最大值)。相应的三条模糊曲线如图 5-7 所示。

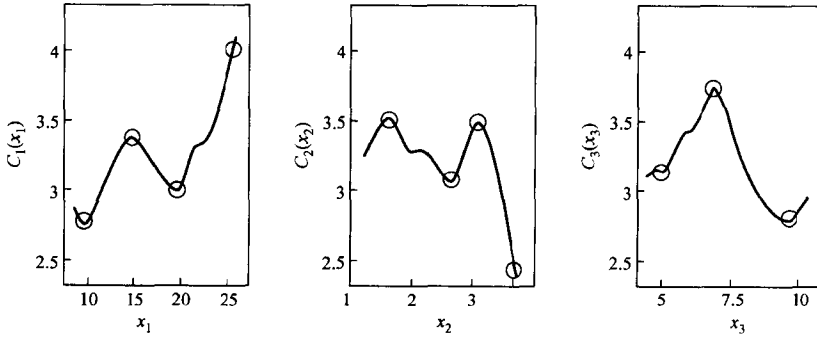


图 5-7 模糊曲线图

设每条曲线中局部最小点和最大点的个数是  $\zeta_n$  (在此例中,  $\zeta_1 = \zeta_2 = 4, \zeta_3 = 3$ ) 则输入空间应划分为  $q$  个模糊区间  $q \geq \max_{n=1 \sim N} (\zeta_n)$  (在此例中  $q \geq 4$ )。然后将每个  $x_n$  分成  $q$  个等间隔区, 每个区的宽度  $\Delta x_n = [\bar{x}_n - x_n]/q$ 。根据每个区的中点上  $C_n(x_n)$  取值的大小对这些区进行排队, 并赋予标号为  $x_n^i, i = 1 \sim q$  ( $i$  值越大, 相应区间中点上的  $C_n(x_n)$  值越大。 $\Delta x_n^q$  是相应于  $C_n(x_n)$  曲线取最大值点的区间;  $x_n^1$  是  $C_n(x_n)$  取最小值点的区间。这些区间的宽度是相同的) 这样可以按下式来计算每一维的  $q$  个隶属函数:

$$\mu_{An}^i(x_n) = \exp \left\{ \left[ - \left( \frac{x_n - x_n^i}{\alpha \Delta x_n} \right)^2 \right]^{\gamma_n^i} \right\}, \quad i = 1 \sim q \quad (5-43)$$

其中  $x_n^i$  是区间  $x_n^i$  的中点,  $\alpha$  可以在  $[0.25, 1.5]$  范围内选择,  $\gamma_n^i$  可以在  $[0.5, 1.0]$  范围内选择。可初步设  $\alpha = 0.5$  及  $\gamma_n^i = 1$ , 在以后进一步的参数学习中予以调整。最后, 输入空间的  $q$  个隶属函数是

$$\mu_A^i(\mathbf{X}) = \prod_{n=1}^N \mu_{An}^i(x_n), \quad i = 1 \sim q \quad (5-44)$$

模糊曲线的建立, 给出了判断输入向量  $\mathbf{X}$  的各维分量  $x_n$  对于形成输出  $y$  的重要性。如果某一维的模糊曲线非常平坦, 这说明它对  $y$  的影响很小, 因此可以将其舍弃。而按局部极值点数量的多少至少可以对输入空间的模糊区个数  $q$  作一个粗糙的初估。当然, 很多细致问题尚待仔细研究。例如 式 (5-42) 的函数  $\varphi_n^b(x_n)$  中, 参数  $\sigma_n$  应取何值还不清楚。

### 5.3.3 结构学习之二 —— 输出空间的模糊划分和模糊推理规则的建立

#### 1. 对输入输出空间分别模糊划分

第一种方案是对于输入向量  $\mathbf{X}$  的空间和输出向量  $\mathbf{Y}$  的空间分别独立地进行模糊区间

划分,可供使用的划分算法已在上列 5.3.2 小节 1~5 各项中给出。假设输入划分为  $q$  个空间,输出划分为  $r$  个空间。这相当于有  $r$  条规则,  $r$  个输出模糊空间相应于  $r$  条规则的结论,每条规则的前提从  $q$  个输入模糊空间中选择,参见上述 5.2.1 小节及式 (5-2)。每一条规则  $R^l$  所涉及的前提模糊空间可用下列方法决定。将训练集中的各对训练向量  $(X_p, Y_p)$  依次输送到模糊推理系统中,如果  $X_p$  对于第  $i(p)$  号输入模糊空间的隶属函数值最大,  $Y_p$  对于第  $l(p)$  号输出模糊空间的隶属函数值最大,则规则  $R^{(p)}$  的前提应包含第  $i(p)$  号输入空间,即式 (5-2) 的前提应包含  $A^{(p)}$ 。当训练集中全部各对向量检验完毕后,即可决定每条规则涉及的前提,前提的个数至少为 1,也可以大于 1。

这种方法还可以用一种在线自适应的方法来实现(见文献[11])。这时训练集不是预先采集好,而是按时序  $t = 1, 2, \dots$  逐对输入  $(X(t), \hat{Y}(t))$ 。起始时输入、输出空间都不作划分,即  $q = r = 0$ 。随着  $t$  的增加,用  $X(t)$  和  $Y(t)$  分别建立输入和输出空间的模糊划分(见上述 5.3.2 小节中 4 项的 (1)~(3))。当  $t = 1$  时,分别用  $X(1) = [x_1(1), \dots, x_N(1)]$  和  $Y(1) = [\hat{y}_1(1), \dots, \hat{y}_M(1)]$  建立输入和输出空间第一个模糊划分区的隶属函数  $\mu_A^1(X)$  和  $\mu_B^1(\hat{Y})$ 。第一条模糊推理规则  $R^1$  的结论为隶属函数  $\mu_B^1(\hat{Y})$ ,而  $\mu_A^1(X)$  是  $R^1$  的前提诸隶属函数中的一个。对于  $t > 1$ , 设已建立了  $q(t-1)$  个前提隶属函数  $\mu_A^i(X)$ ,  $i = 1 \sim q(t-1)$  以及  $r(t-1)$  个输出(即结论)隶属函数  $\mu_B^l(\hat{Y})$ ,  $l = 1 \sim r(t-1)$  (这相应于建立了  $r(t-1)$  条规则),设  $J(t)$  用式 (5-23) 计算且设

$$\hat{J}(t) = \arg \max_{l=1 \sim r(t-1)} (\mu_B^l(\hat{Y}(t)))$$

若  $\mu_A^{J(t)}(X(t)) \geq \mu_T(t)$  且  $\mu_B^{\hat{J}(t)}(\hat{Y}(t)) \geq \hat{\mu}_T(t)$  ( $\mu_T(t)$  与  $\hat{\mu}_T(t)$  相似,是一个随  $t$  增加而渐降的门限函数,若  $\mu_T(t)$  愈贴近于 1 时输入空间划分得越细,而  $\hat{\mu}_T(t)$  越贴近 1 时输出空间划分得越细。当  $t = 1$  时,  $\mu_T(1)$  和  $\hat{\mu}_T(1)$  皆小于 1 但比较接近于 1) 则输入和输出都无须增加新模糊空间,因而无须增加新规则。这时所需做的事只是将  $\mu_A^{J(t)}(X)$  包含于第  $\hat{J}(t)$  条规则的前提之中(如本来已包含,则无需做任何事)。若  $\mu_A^{J(t)}(X(t)) < \mu_T(t)$  且  $\mu_B^{\hat{J}(t)}(\hat{Y}(t)) \geq \hat{\mu}_T(t)$  则增加一个输入模糊空间,令  $q(t) = q(t-1) + 1$  且  $\mu_A^{q(t)}(X)$  可用  $X(t)$  构造。而无须增加输出空间及新规则。这时可以将  $\mu_A^{q(t)}(X)$  包含于第  $\hat{J}(t)$  条规则的前提之中。若  $\mu_A^{J(t)}(X(t)) \geq \mu_T(t)$  且  $\mu_B^{\hat{J}(t)}(\hat{Y}(t)) < \hat{\mu}_T(t)$ , 则输入模糊空间无须增加,即  $q(t) = q(t-1)$ 。而输出模糊空间及规则数都增加 1,即  $r(t) = r(t-1) + 1$ 。新增的模糊隶属函数  $\mu_B^{r(t)}(\hat{Y})$  可用  $\hat{Y}(t)$  来构造,它即是新增加的第  $r(t)$  条规则的结论隶属函数。 $\mu_A^{J(t)}(X)$  则是这条规则的前提隶属函数之一。若  $\mu_A^{J(t)}(X(t)) < \mu_T(t)$  且  $\mu_B^{\hat{J}(t)}(\hat{Y}(t)) < \hat{\mu}_T(t)$  则输入和输出模糊空间及规则数皆增加 1,即  $q(t) = q(t-1) + 1$  且  $r(t) = r(t-1) + 1$ 。 $\mu_A^{q(t)}(X)$  和  $\mu_B^{r(t)}(\hat{Y})$  分别用  $X(t)$  和  $\hat{Y}(t)$  构造且前者是后者的前提之一。

## 2. 对输入、输出合并后的空间模糊划分

第二种方法是将输入  $X$  和输出  $Y$  合成一个  $N + M$  维向量  $Z$ , 相应的训练集  $(X_p, Y_p)$ ,

$p = 1 \sim P$  将成为  $Z_p, p = 1 \sim P$ 。接着针对  $Z$  向量空间进行模糊划分 (采用上述 5.3.2 小节 1 至 5 中所列的各种方法) 假设分成了  $q$  个模糊空间。由于  $Z$  包含  $X$  和  $Y, X$  和  $Y$  相应地各自分成了  $q$  个模糊空间, 每个空间有自身的隶属函数且一个  $X$  模糊空间与一个  $Y$  模糊空间具有固定对应关系。这时系统的规则数  $r$  即等于  $q$ , 每条规则只有一个前提且前提与结论之间的对应关系在对  $Z$  作模糊划分时已确定下来。

### 3. 用切割法或模糊曲线法进行模糊划分

在采用 5.3.2 小节中 6、7 所述的切割法或模糊曲线法对输入空间进行模糊划分时, 一个前提只对应一个结论, 即系统的规则数  $r$  即等于输入空间的模糊区间数  $q$ 。在用切割法时 即可用式 (5-40) 求得的各  $C_{b_i}$  作为 0 阶或 1 阶 TSK 模型中系数  $b'_0$  的初估值 (见式 (5-5) 或式 (5-8))。在采用模糊曲线法时 可将输出  $y$  的全部变化区间分成  $q$  个等间隔 若  $y$  的上、下限分别为  $\bar{y}, y$  则每个间隔的宽度为  $\Delta y = (\bar{y} - y)/q$ 。按  $y$  值从小到大将各  $\Delta y$  编号为  $y^i, i = 1 \sim q$  设各  $y^i$  的中心为  $y'_0$ 。系统的  $q$  个前提隶属函数  $\mu'_A(X)$  (见式 (5-44)) 中每一个与相应编号的输出  $y^i$  相对应, 即每条规则只有一个前提对应着一个结论,  $r = q$ 。这时各条规则结论中的  $y'_0$  即可作为 0 阶或 1 阶 TSK 模型中系数  $b'_0$  的初估值。

### 5.3.4 采用 0 阶 TSK 模型时的参数学习 (BP 算法)

在结构学习阶段, 决定了前提数  $q$  规则数  $r$  此外输入向量  $X$  的维数  $N$  和输出向量  $Y$  的维数  $M$  是已知的 (为简化 设  $M = 1$ )。如果用式 (5-21) 表示前提隶属函数  $\mu'_A(X)$  则每个函数包含  $3N$  个参数  $M'_n, \sigma'_n, \gamma'_n, n = 1 \sim N$ 。 $q$  个前提共含  $3qN$  个参数。此外  $r$  条规则结论可用下列隶属函数表示:

$$\mu'_B(y) = \exp \left\{ - \left( \frac{y - b'_0}{\sigma'_0} \right)^2 \right\}, \quad l = 1 \sim r \quad (5-45)$$

这样 共含  $2r$  个参数。这些参数在结构学习时已初步估计出, 但尚需予以精细调节。

参数精细调节的最重要方法仍是 BP 算法 (其基础是最陡下降算法)。对于 0 阶 TSK 模型 各参数只需采取 BP 算法予以细调。与普通神经网络中用 BP 算法调节参数不同的方面是, 普通神经网络中网络规模和各参数初值的设置是盲目的, 因此由 BP 算法搜索到的参数值可能使目标函数达到某个不佳的局部最小点且网络规模不是最佳的。而 FNN 中经过结构学习后, 无论网络规模还是网络中初步确定的各参数都已接近于最佳, 这时经过 BP 算法进行学习就可以较快地达到最佳参数值。

BP 算法可以采取对整体训练集求目标函数并予以优化的批处理方式, 也可采取针对逐个训练样本求目标函数并予以优化的随机梯度方式。为简化起见, 这里只介绍针对 MISO 系统的随机梯度参数优化算法。

设按节拍  $k = 0, 1, 2, \dots$  逐对输入训练样本  $\{X(k), \hat{y}(k)\}$ 。FNN 的输出  $y(k)$  按式 (5-6) 计算:

$$y(k) = \frac{\sum_{l=1}^r b'_0(k) \sigma'_0(k) \mu^l(k)}{\sum_{l=1}^r \sigma'_0(k) \mu^l(k)} \quad (5-46)$$

其中  $\mu^l(k)$  表示  $\mu^l(\mathbf{X}(k))$ ,  $b_0^l(k)$  和  $\sigma_0^l(k)$  表示随  $k$  而变化的参数  $b_0^l$  和  $\sigma_0^l$ 。将结构学习阶段求得的参数作为此处的初值  $b_0^l(0)$  和  $\sigma_0^l(0)$ 。设置随机梯度算法目标函数  $E(k)$  如下：

$$E(k) = [\hat{y}(k) - y(k)]^2 \quad (5-47)$$

对于每一  $k$  值用最陡下降算法求各个参数，使得参数改变后  $E(k)$  得以下降。下面先讨论  $b_0^l$  和  $\sigma_0^l$  这一对参数，然后再讨论前提各隶属函数中的参数。

参数  $b_0^l$  和  $\sigma_0^l$  的调整计算公式是

$$\left. \begin{aligned} b_0^l(k+1) &= b_0^l(k) + \Delta b_0^l(k), \quad \Delta b_0^l(k) = -\alpha_b(k) \frac{\partial E(k)}{\partial b_0^l(k)} \\ \sigma_0^l(k+1) &= \sigma_0^l(k) + \Delta \sigma_0^l(k), \quad \Delta \sigma_0^l(k) = -\alpha_\sigma(k) \frac{\partial E(k)}{\partial \sigma_0^l(k)} \end{aligned} \right\} \quad (5-48)$$

其中  $\alpha_b(k)$ 、 $\alpha_\sigma(k)$  为随  $k$  的增加而渐降的非负步幅系数。通过简单运算，即可求得

$$\left. \begin{aligned} -\frac{\partial E(k)}{\partial b_0^l(k)} &= 2[\hat{y}(k) - y(k)] \frac{\partial y(k)}{\partial b_0^l(k)} = 2[\hat{y}(k) - y(k)] \frac{\sigma_0^l(k) \mu^l(k)}{\sum_{u=1}^r \sigma_0^u(k) \mu^u(k)} \\ -\frac{\partial E(k)}{\partial \sigma_0^l(k)} &= 2[\hat{y}(k) - y(k)] \frac{\partial y(k)}{\partial \sigma_0^l(k)} = 2[\hat{y}(k) - y(k)] \times \\ &\quad \frac{\mu^l(k) [b_0^l(k) \sum_{u=1}^r \sigma_0^u(k) \mu^u(k) - \sum_{u=1}^r b_0^u(k) \sigma_0^u(k) \mu^u(k)]}{\left[ \sum_{u=1}^r \sigma_0^u(k) \mu^u(k) \right]^2} \end{aligned} \right\} \quad (5-49)$$

为计算各前提隶属函数中各参数的调整量，首先将输入为  $\mathbf{X}(k) = [x_1(k), \dots, x_N(k)]$  时的  $q$  个前提隶属函数取值列出如下（见式 5-21）。为简化起见，设  $\gamma_n \equiv 1$ （即参数  $\gamma_n$  作为一个不进行调整的恒值），即得

$$\mu_A^i(k) = \mu_A^i(\mathbf{X}(k)) = \prod_{n=1}^N \exp \left[ - \left( \frac{x_n(k) - M_n^i(k)}{\sigma_n^i(k)} \right)^2 \right], \quad i = 1 \sim q \quad (5-50)$$

再利用式 5-3) 可知

$$\mu^l(k) = \mu^l(\mathbf{X}(k)) = \mu_A^{i^*(l)}(\mathbf{X}(k)) = \max[\mu_A^{i_1^{(l)}}(\mathbf{X}(k)), \dots, \mu_A^{i_p^{(l)}}(\mathbf{X}(k))] \quad (5-51)$$

其中  $i^*(l) \in \{i_1(l), \dots, i_p(l)\} \subset \{1, 2, \dots, q\}$ 。

这样按节拍  $k = 0, 1, 2, \dots$  输入  $\{\mathbf{X}(k), \hat{y}(k)\}$  且用式 5-46) 计算  $y(k)$  用式 5-49) 计算目标函数  $E(k)$  且令  $M_n^i(0), \sigma_n^i(0), i = 1 \sim q, n = 1 \sim N$ ，为结构学习阶段得到的参数。这样可以导出各  $M_n^i$  和  $\sigma_n^i$  的调整计算公式，如下：

$$\left. \begin{aligned} M_n^i(k+1) &= M_n^i(k) + \Delta M_n^i(k), \quad i = 1 \sim q, n = 1 \sim N \\ \sigma_n^i(k+1) &= \sigma_n^i(k) + \Delta \sigma_n^i(k), \quad i = 1 \sim q, n = 1 \sim N \end{aligned} \right\} \quad (5-52)$$

$$\left. \begin{aligned} \Delta M_n^i(k) &= -\alpha_m(k) \frac{\partial E(k)}{\partial M_n^i(k)} = 2\alpha_m(k) [\hat{y}(k) - y(k)] \frac{\partial y(k)}{\partial M_n^i(k)} \\ \Delta \sigma_n^i(k) &= -\alpha_\sigma(k) \frac{\partial E(k)}{\partial \sigma_n^i(k)} = 2\alpha_\sigma(k) [\hat{y}(k) - y(k)] \frac{\partial y(k)}{\partial \sigma_n^i(k)} \end{aligned} \right\} \quad (5-53)$$

其中  $\alpha_m(k), \alpha_\sigma(k)$  是随着  $k$  的增加逐渐下降的非负步幅函数，当  $k = 0$  时它们的取值略小于 1，当  $k$  时，它们渐趋于 0。

根据式 (5-46)、(5-50) 和 (5-51), 各  $M_n^i$  和  $\sigma_n^i$  对输出  $y$  的作用必通过  $\mu^i(k)$ ,  $= 1 \sim r$ , 所以可建立如下的全微分公式:

$$\left. \begin{aligned} \frac{\partial y(k)}{\partial M_n^i(k)} &= \sum_{l=1}^r \frac{\partial y(k)}{\partial \mu^l(k)} \cdot \frac{\partial \mu^l(k)}{\partial M_n^i(k)} \\ \frac{\partial y(k)}{\partial \sigma_n^i(k)} &= \sum_{l=1}^r \frac{\partial y(k)}{\partial \mu^l(k)} \cdot \frac{\partial \mu^l(k)}{\partial \sigma_n^i(k)} \end{aligned} \right\} \quad (5-54)$$

利用式 (5-46) 可以求得

$$\frac{\partial y(k)}{\partial \mu^i(k)} = \frac{\sigma_0^i(k) \left[ b_0^i(k) \sum_{u=1}^r \sigma_0^u(k) \mu^u(k) - \sum_{u=1}^r b_0^u(k) \sigma_0^u(k) \mu^u(k) \right]}{\left[ \sum_{u=1}^r \sigma_0^u(k) \mu^u(k) \right]^2} \quad (5-55)$$

再利用式 (5-50) 和 (5-51) 可以求得

$$\left. \begin{aligned} \frac{\partial \mu^i(k)}{\partial M_n^i(k)} &= \mu_A^{i*}(\mathbf{X}(k)) \frac{2[x_n(k) - M_n^i(k)]}{[\sigma_n^i(k)]^2}, & i = i^*(l), \quad n = 1 \sim N \\ \frac{\partial \mu^i(k)}{\partial M_n^i(k)} &= 0, & i \neq i^*(l) \end{aligned} \right\} \quad (5-56)$$

以及

$$\left. \begin{aligned} \frac{\partial \mu^i(k)}{\partial \sigma_n^i(k)} &= \mu_A^{i*}(\mathbf{X}(k)) \frac{2[x_n(k) - M_n^i(k)]^2}{[\sigma_n^i(k)]^3}, & i = i^*(l), \quad n = 1 \sim N \\ \frac{\partial \mu^i(k)}{\partial \sigma_n^i(k)} &= 0, & i \neq i^*(l) \end{aligned} \right\} \quad (5-57)$$

将式 (5-55)、(5-57) 代入式 (5-54) 再将式 (5-54) 代入式 (5-53) 便可求得各参数的调整量。

### 5.3.5 采用 1 阶 TSK 模型时的参数学习 (OLS 算法)

1 阶 TSK 模型在完成复杂的函数映射时可以用少量的规则达到较高的函数逼近精度, 但是每一条推理规则  $R^l$  的结论部分须确定  $N+1$  个参数  $b_0^l, b_1^l, \dots, b_N^l$  (见式 (5-4) 和式 (5-8))。确定这些参数的第一种方案是用 BP 算法 (包括前提隶属函数中各参数的确定在内, 其算法可参照 5.3.4 小节所述方法导出, 不再赘述)。这一方案的缺点是: (1) 将前提与结论参数放在一起进行优化, 使 BP 算法的计算量增加很多。而且在结构学习阶段并未确定诸  $b_1^l \sim b_N^l$  之值, 因此在进行 BP 学习时只能设其初值为随机值。这样既使学习效率降低又可能陷入某个不佳的局部最小值点。(2) 在各个  $b_1^l \sim b_N^l$  中, 有些对输出的影响很大, 而有些影响很小。BP 算法无法对此进行区分, 将所有参数都纳入计算, 其结果是既不能保留最重要的系数而删除不重要的系数 (即不能实现一个结构简单的网络) 又学习效率低。因此, 现在研究的重点放在第二种方案上, 即将前提参数的学习和结论参数的学习分开, 前者仍用 BP 算法, 后者直接进行优化。因为  $b_0^l \sim b_N^l$  是一个线性组合式的系数, 所以可采取高效的学习算法并避免陷入局部极小点。在实现时, 可以令前提参数的 BP 算法学习和结论参数的学习交替运行。这一小节将介绍用 OLS 算法的结论参数学习 (OLS 是 orthogonal least square 的缩写)。

首先 将 1 阶 TSK 模型中输入  $\mathbf{X}$  至输出  $y$  的映射关系 式 (5-8) , 改写为下列形式 :

$$y = \sum_{l=1}^r (b_0^l + b_1^l x_1 + \cdots + b_N^l x_N) \gamma^l(\mathbf{X}) \quad (5-58)$$

其中

$$\gamma^l(\mathbf{X}) = \frac{\mu^l(\mathbf{X})}{\sum_{u=1}^r \mu^u(\mathbf{X})}$$

若  $\gamma^l(\mathbf{X}), l = 1 \sim r$  为已知 , 训练集为  $(\mathbf{X}_m, \hat{y}_m), m = 1 \sim M$ 。对于输入  $\mathbf{X}_m = [x_{m1}, \cdots, x_{mN}]$ ,  $\hat{y}_m$  为系统的理想输出 , 实际输出  $y_m$  可用下式计算 :

$$y_m = \sum_{l=1}^r (b_0^l + b_1^l x_{m1} + \cdots + b_N^l x_{mN}) \gamma_m^l \quad (5-59)$$

其中  $\gamma_m^l = \gamma^l(\mathbf{X}_m)$ 。OLS 算法的目的是求  $r$  组最佳系数  $(b_0^l, b_1^l, \dots, b_N^l), l = 1 \sim r$  使得

$\sum (\hat{y}_m - y_m)^2$  达到最小。为此将式 (5-59) 改写为下列形式 :

$$y_m = \mathbf{F}_m \boldsymbol{\theta}^T \quad (5-60)$$

其中  $\boldsymbol{\theta}$  是一个  $(N+1)r$  维行向量 其元素为  $\theta_i, i = 1 \sim (N+1)r$  , 可由下列对照关系确定 :

$$\begin{aligned} \boldsymbol{\theta} &= [\theta_{11}, \theta_{12}, \dots, \theta_r, \theta_{r+1}, \theta_{r+2}, \dots, \theta_{2r}, \dots, \theta_{Nr+1}, \theta_{Nr+2}, \dots, \theta_{(N+1)r}] \\ &= [b_0^1, b_0^2, \dots, b_0^r, b_1^1, b_1^2, \dots, b_1^r, \dots, b_N^1, b_N^2, \dots, b_N^r] \end{aligned}$$

$\mathbf{F}_m$  也是一个  $(N+1)r$  维行向量 其元素为  $f_{mi}, i = 1 \sim (N+1)r$  , 可由下列对照关系确定 :

$$\begin{aligned} \mathbf{F}_m &= [f_{m1}, f_{m2}, \dots, f_{mr}, f_{m,r+1}, f_{m,r+2}, \dots, f_{m,2r}, \dots, f_{m,Nr+1}, f_{m,Nr+2}, \dots, f_{m,(N+1)r}] \\ &= [\gamma_m^1, \gamma_m^2, \dots, \gamma_m^r, \gamma_m^1 x_{m1}, \gamma_m^2 x_{m1}, \dots, \gamma_m^r x_{m1}, \dots, \gamma_m^1 x_{mN}, \gamma_m^2 x_{mN}, \dots, \gamma_m^r x_{mN}] \end{aligned}$$

对于任何输入  $\mathbf{X}_m$  和已知函数  $\mu^l(\mathbf{X}), l = 1 \sim r$  就可以求得  $\mathbf{F}_m$ 。这样 式 (5-60) 可写成

$$y_m = \sum_{i=1}^{(N+1)r} f_{mi} \theta_i \quad (5-61)$$

通过下列正交化程序可以求得一组  $\theta_i$  的最佳估值 使得  $\sum (\hat{y}_m - y_m)^2$  达到最小 [18,19,20]

正交化是指首先求得  $\mathbf{F}_m, m = 1 \sim M$  所张成的  $(N+1)r$  维空间中的  $(N+1)r$  个正交归一基向量 (维数为  $(N+1)r$  的列向量) 它们构成一个  $[(N+1)r] \times [(N+1)r]$  维正交变换方阵  $\mathbf{Q}, \mathbf{Q}\mathbf{Q}^T = \mathbf{I}$  ( $\mathbf{I}$  是一个  $[(N+1)r] \times [(N+1)r]$  维单位矩阵) 这样式 (5-60) 可以改写为

$$y_m = \mathbf{F}_m \mathbf{Q} \mathbf{Q}^T \boldsymbol{\theta}^T = \mathbf{W}_m \mathbf{G}^T = \sum_{i=1}^{(N+1)r} w_{mi} g_i \quad (5-62)$$

其中

$$\begin{aligned} \mathbf{F}_m \mathbf{Q} &= \mathbf{W}_m = [w_{m1}, w_{m2}, \dots, w_{m,(N+1)r}] \\ \mathbf{Q}^T \boldsymbol{\theta}^T &= \mathbf{G}^T = [g_1, g_2, \dots, g_{(N+1)r}]^T \end{aligned}$$

各  $w_{mi}$  可通过下列计算程序求得。

(1) 对于  $i = 1$  有

$$w_{m1} = f_{m1}, \quad m = 1 \sim M \quad (5-63)$$

(2) 对于  $i = 2 \sim (N+1)r$  有



$$w_{mi} = f_{mi} - \sum_{j=1}^{i-1} \alpha_{ji} w_{mj}, \quad m = 1 \sim M \quad (5-64)$$

其中

$$\alpha_{ji} = \frac{\sum_{m=1}^M w_{mj} f_{mi}}{\sum_{m=1}^M w_{mj}^2}, \quad j = 1, 2, \dots, i-1 \quad (5-65)$$

在求得各  $w_{mi}$  后, 可以求得使  $\frac{1}{M} \sum_{m=1}^M (\hat{y}_m - y_m)^2$  达到最小的诸  $g_i$  的最佳估值  $\hat{g}_i$  如下:

$$\hat{g}_i = \frac{\sum_{m=1}^M w_{mi} \hat{y}_m}{\sum_{m=1}^M w_{mi}^2}, \quad i = 1 \sim (N+1)r \quad (5-66)$$

由各  $\hat{g}_i$  可求得诸  $\theta_i$  的最佳估值  $\hat{\theta}_i$  如下:

(1) 对于  $i = (N+1)r$  有

$$\hat{\theta}_{(N+1)r} = \hat{g}_{(N+1)r} \quad (5-67)$$

(2) 对于  $i = (N+1)r-1, (N+1)r-2, \dots, 1$  有

$$\hat{\theta}_i = \hat{g}_i - \sum_{q=i+1}^{(N+1)r} \alpha_{iq} \hat{\theta}_q \quad (5-68)$$

$\alpha_{iq}$  的定义见式 (5-65)。

最后, 可以算出在式 (5-62) 中每当  $g_i$  值由 0 变成  $\hat{g}_i$  时,  $\frac{1}{M} \sum_{m=1}^M (\hat{y}_m - y_m)^2$  将下降  $[\Delta e]_i$ :

$$[\Delta e]_i = \frac{\hat{g}_i^2 \sum_{m=1}^M w_{mi}^2}{\sum_{m=1}^M \hat{y}_m^2}, \quad i = 1 \sim (N+1)r \quad (5-69)$$

$[\Delta e]_i$  值越大 表明相应的  $\hat{g}_i$  项对输出的影响越大。可以设置一个阈值  $\rho$  对于  $[\Delta e]_i \geq \rho$  的那些  $\hat{g}_i$  予以保留 而  $[\Delta e]_i < \rho$  的那些  $\hat{g}_i$  予以删除。这样就可减少结论部分中的参数。

以上删除过程需待所有  $(N+1)r$  个  $[\Delta e]_i$  都算出来以后才能实施。采取下列运算方案可以在选出  $M$  个  $[\Delta e]_i$  最大的  $\hat{g}_i$  后即停止计算, 从而能够有效地节约计算量。这种算法称为 Gram-Schmidt 正交化算法。

按照节拍  $k = 1, 2, \dots$  进行迭代计算, 依次求出  $w_{mi}^{(k)}, i = 1 \sim (N+1)r, m = 1 \sim M$ , 并且求出  $\hat{g}_i^{(k)}$  和  $[\Delta e]_i^{(k)}, i = 1 \sim (N+1)r$ 。算法如下所列。

(1) 对于  $k = 1$

令  $w_{mi}^{(1)} = f_{mi}, \quad i = 1 \sim (N+1)r, \quad m = 1 \sim M$   
 求出

$$\hat{g}_i^{(1)} = \frac{\sum_{m=1}^M w_{mi}^{(1)} \hat{y}_m}{\sum_{m=1}^M [w_{mi}^{(1)}]^2}, \quad [\Delta e]_i^{(1)} = \frac{[\hat{g}_i^{(1)}]^2 \sum_{m=1}^M [w_{mi}^{(1)}]^2}{\sum_{m=1}^M \hat{y}_m^2}, \quad i = 1 \sim (N+1)r$$

设

$$[\Delta e]_a^{(1)} = \max_{i=1 \sim (N+1)r} \{[\Delta e]_i^{(1)}\}$$

则令

$$w_{m1} = w_{ma}^{(1)} = f_{ma}, \quad m = 1 \sim M$$

且

$$\hat{g}_1 = \hat{g}_a^{(1)}, \quad [\Delta e]_1 = [\Delta e]_a^{(1)}$$

(2) 对于  $k = 2$

$$\text{令 } w_{mi}^{(2)} = f_{mi} - \alpha_{1i} w_{m1}, \quad i = 1 \sim (N+1)r, \quad i \neq a, \quad m = 1 \sim M$$

其中

$$\alpha_{1i} = \frac{\sum_{m=1}^M w_{m1} f_{mi}}{\sum_{m=1}^M w_{m1}^2} \quad (\text{式(5-65)})$$

求出

$$\hat{g}_i^{(2)} = \frac{\sum_{m=1}^M w_{mi}^{(2)} \hat{y}_m}{\sum_{m=1}^M [w_{mi}^{(2)}]^2}, \quad [\Delta e]_i^{(2)} = \frac{[\hat{g}_i^{(2)}]^2 \sum_{m=1}^M [w_{mi}^{(2)}]^2}{\sum_{m=1}^M \hat{y}_m^2}, \quad i = 1 \sim (N+1)r, \quad i \neq a$$

设

$$[\Delta e]_b^{(2)} = \max_{i=1 \sim (N+1)r, i \neq a} \{[\Delta e]_i^{(2)}\}$$

则令

$$w_{m2} = w_{mb}^{(2)} = f_{mb} - \alpha_{1b} w_{m1}, \quad m = 1 \sim M$$

且

$$\hat{g}_2 = \hat{g}_b^{(2)}, \quad [\Delta e]_2 = [\Delta e]_b^{(2)}$$

(3) 进行下列检验:  $1 - \sum_{i=1}^k [\Delta e]_i < \rho$  ? ( $\rho$  是一个阈值)

若回答为非 则令  $k = k + 1$  再转而进行(2)项规定的类似运算。

若回答为是 则运算结束 令  $M = k$ 。M 即为系数结论部分所涉及的系数的个数。借助于式(5-67)和(5-68)可求出这些系数。

### 5.3.6 采用 1 阶 TSK 模型时的参数学习 (RLSE 算法)

RLSE 是 recursive least square estimation 的缩写。RLSE 算法的目的与 OLS 算法相同, 即对于已知前提隶属函数  $\gamma^l(X), l = 1 \sim r$  的条件下, 针对训练集  $(X_m, \hat{y}_m), m = 1 \sim M$  求  $r$  组最佳系数  $b_0^l, b_1^l, \dots, b_N^l, l = 1 \sim r$  使得按式(5-59)算出的实际输出  $y_m$  与理想输出  $\hat{y}_m$  之间的误差平方和  $\sum_{m=1}^M (y_m - \hat{y}_m)^2$  达到最小。与 5.3.5 小节相同, 首先将  $y_m$  表示为 (见式(5-60)和式(5-61))

$$y_m = F_m \theta^T = \sum_{i=1}^{(N+1)r} f_{mi} \theta_i, \quad m = 1 \sim M$$

将这  $M$  个方程并成如下一个矩阵方程:

$$\underbrace{\begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1,(N+1)r} \\ f_{21} & f_{22} & \cdots & f_{2,(N+1)r} \\ \cdots & \cdots & \cdots & \cdots \\ f_{M1} & f_{M2} & \cdots & f_{M,(N+1)r} \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{(N+1)r} \end{bmatrix}}_{\theta^T} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix}}_Y \quad (5-70)$$

易于导出

$$\sum_{m=1}^M (y_m - \hat{y}_m)^2 = \| \mathbf{Y} - \hat{\mathbf{Y}} \|^2 = \| \Phi \boldsymbol{\theta}^T - \hat{\mathbf{Y}} \|^2 \quad (5-71)$$

其中  $\hat{\mathbf{Y}} = [\hat{y}_1, \hat{y}_2, \dots, \hat{y}_M]^T$ ,  $\mathbf{Y} = [y_1, y_2, \dots, y_M]^T$ ,  $\Phi$  是一个  $M \times (N+1)r$  维矩阵, 且可表示为

$$\Phi = \begin{bmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \\ \vdots \\ \mathbf{F}_M \end{bmatrix}$$

现在的问题归结为, 在已知  $\Phi$  和  $\mathbf{Y}$  的条件下 求一最佳  $\boldsymbol{\theta}$  使式 (5-71) 右侧达到最小。该最佳解用  $\boldsymbol{\theta}$  表示。这就是信号处理和自适应滤波等领域中常遇到的线性回归 (linear regression) 问题。如果  $\Phi^T \Phi$  是非奇的, 则可求得最佳解为

$$\hat{\boldsymbol{\theta}}^T = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{Y} \quad (5-72)$$

这涉及一个高维矩阵求逆的问题, 计算量非常大, 而且当  $\Phi^T \Phi$  为奇时无法求解。在上一小节中采用正交化的方法解决此问题。在这一节用序贯迭代计算方法求此问题的解<sup>[12,21]</sup>

设  $\mathbf{S}_k$  是一个  $(N+1)r \times (N+1)r$  维方阵, 且设

$$\mathbf{S}_0 = \gamma \mathbf{I} \quad (5-73)$$

其中  $\gamma$  是一个充分大的正数,  $\mathbf{I}$  是一个  $(N+1)r \times (N+1)r$  维单位阵。按照节拍  $k = 0, 1, 2, \dots, M$  可以进行下列序贯迭代计算:

$$\mathbf{S}_{k+1} = \mathbf{S}_k - \frac{\mathbf{S}_k \mathbf{F}_{k+1}^T \mathbf{F}_{k+1} \mathbf{S}_k}{1 + \mathbf{F}_{k+1}^T \mathbf{S}_k \mathbf{F}_{k+1}}, \quad k = 0, 1, \dots, M-1 \quad (5-74)$$

注意其中  $\mathbf{F}_{k+1} = [f_{(k+1),1}, f_{(k+1),2}, \dots, f_{(k+1),(N+1)r}]$ 。

在进行上列迭代计算的每一步同时进行下列迭代计算:

$$\boldsymbol{\theta}_{k+1}^T = \boldsymbol{\theta}_k^T + \mathbf{S}_{k+1} \mathbf{F}_{k+1}^T (\hat{y}_{k+1} - \mathbf{F}_{k+1} \boldsymbol{\theta}_k^T) \quad k = 0, 1, \dots, M-1 \quad (5-75)$$

其中迭代计算初值为  $\boldsymbol{\theta} = \mathbf{0}$ 。此迭代计算的最终值  $\boldsymbol{\theta}_M$  即是所求的解  $\boldsymbol{\theta}$  即

$$\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}_M \quad (5-76)$$

## 5.4 基于 Fuzzy ART 的 FNN 和 Simpson 网络

### 5.4.1 Fuzzy ART 和 Simpson 的 MIN-MAX 网络

在第 3 章已介绍了 ART 的基本原理 由于 ART 对模仿人的认知过程 概念建立、记忆机制等 具有独到之处 所以一直受到重视。但是最早提出的 ART1, ART2, ART3 等网络过于复杂 因此难以得到实际应用。从 20 世纪 90 年代初期开始 两种高效 ART 网络提了出来, 一种是 ART2A 另一种就是 Fuzzy ART 前者见第 5 章的介绍, 后者见文献[22]、[23]。Fuzzy ART 继承了 ART 的思路, 融合了模糊推理的方法, 既有强在线自适应能

力，良好的类别或聚类分辨率，又有很好的可操作性（学习和运作都简易可行），所以近几年来受到研究界的重视。

Fuzzy ART 也是一个 5 层 FNN 它的各层的功能与 5.3.1 小节所述网络结构类似。其突出特点如下所列。

(1) 采取梯形的一维隶属函数  $\mu_{A_n}^i(x_n)$  在网络第 2 层计算)，其计算公式见式 5-18)。每一个  $\mu_{A_n}^i(x_n)$  用最小点（即 MIN 点或左端点） $U_n^i$ 、最大点（MAX 点或右端点） $V_n^i$  和衰减系数  $\gamma_n^i$  描述。对于  $N$  维输入向量  $\mathbf{X} = [x_1, \dots, x_N]$  其对于第  $i$  个输入模糊空间的隶属度  $\mu_A^i(\mathbf{X})$  的计算公式见式 5-16)（在网络第 3 层计算）。 $N$  对（最小点、最大点）—— $(U_n^i, V_n^i)$ ， $n = 1 \sim N$  描述了  $N$  维空间中的一个超立方体（hyperbox）。当  $\mathbf{X}$  在立方体内时  $\mu_A^i(\mathbf{X}) = 1$ 。当  $\mathbf{X}$  在立方体外时  $\mu_A^i(\mathbf{X}) < 1$  且  $\mathbf{X}$  距立方体越远时  $\mu_A^i(\mathbf{X})$  越小。一般取  $\gamma_n^i$  为一恒定值  $\gamma$ ， $\gamma$  越大时  $\mu_A^i(\mathbf{X})$  在超立方体外衰减越快。网络的第 4 层完成  $r$  条规则的结论部分隶属函数  $\mu^l(\mathbf{X})$ ， $l = 1 \sim r$  的计算（见式 5-3）。网络的第 5 层可根据不同的应用目的进行设计。对于分类或聚类应用， $r$  条规则的结论对应于  $r$  个类别或  $r$  个聚类区。如果只要求系统进行软判决，则第 5 层无须作任何处理，直接给出  $r$  个类别或聚类区的隶属函数值即可。如果要进行硬判决，则令诸  $\mu^l(\mathbf{X})$  中最大者的相应类别或聚类区输出端等于 1 其他输出端等于 0。对于函数逼近，若输出是  $M$  维向量  $\mathbf{Y} = [y_1, \dots, y_M]$  那么每一条规则的输出隶属函数  $\mu_B^l(\mathbf{Y})$ ， $l = 1 \sim r$  也可以用  $r$  个  $M$  维空间超立方格定义，后者用  $M$  对（最小点、最大点）—— $(U_m^l, V_m^l)$ ， $m = 1 \sim M$ ，以及衰减系数  $\gamma^l$  描述。如果采用 0 阶 TSK 模型，在第 5 层可以用下列公式计算出系统的  $M$  维输出向量  $\mathbf{Y}$  此式为式 5-5) 的推广)：

$$\mathbf{Y} = \frac{\sum_{l=1}^r \mathbf{B}_0^l \mu^l(\mathbf{X})}{\sum_{l=1}^r \mu^l(\mathbf{X})} \quad (5-77)$$

其中

$$\begin{aligned} \mathbf{B}_0^l &= [b_{01}^l, b_{02}^l, \dots, b_{0M}^l], \quad l = 1 \sim r \\ b_{0m}^l &= \frac{1}{2} (U_m^l + V_m^l), \quad m = 1 \sim M \end{aligned}$$

(2) 网络的学习采取在线自适应方式（见前述 5.3.3 小节的 1）。Fuzzy ART 的在线自适应学习算法与 5.3.3 小节 1 所述算法的区别在于，对于后者，当时间节拍  $k$  输入训练样本对  $(\mathbf{X}(k), \mathbf{Y}(k))$  且  $\mathbf{X}(k)$  或  $\hat{\mathbf{Y}}(k)$  对原建立的某个输入（或输出）模糊区间的隶属度足够高时，原建模糊区不作任何更改，否则建立一个新的模糊区。对于前者，则采用“谐振检验法”判断  $\mathbf{X}(k)$ （或  $\hat{\mathbf{Y}}(k)$ ）与某个已建立模糊区之间的相似度是否达到了某个阈值，若达到了则用此  $\mathbf{X}(k)$  或  $\hat{\mathbf{Y}}(k)$  对原建区间作适当修正（这称为进入自适应谐振状态），若一个也没有达到，则需建立一个新模糊区。可以看到，若环境发生变化时，5.3.3 小节的 1 所述方案中已建立的模糊区不可能随新的样本  $\mathbf{X}(k)$ （或  $\hat{\mathbf{Y}}(k)$ ）而改变，因而不可能真正实现对环境变化的自适应；而 Fuzzy ART 能做到这一点。

这一节还将叙述一种与 Fuzzy ART 关系密切的 FNN，这就是 Simpson MIN-MAX 网络<sup>[24,25]</sup>。Simpson 网络也采用超立方体式隶属函数和 5 层网络结构，其运行方式与 Fuzzy

ART相似，而学习方法不一样。对于时间  $k$  输入的  $\mathbf{X}(k)$  或  $\hat{\mathbf{Y}}(k)$ ，Simpson 网络根据其对已建各模糊区的隶属度，从大到小进行可扩张性检验，每个模糊区的超立方体最大体积定为  $\theta$ 。如果为包含  $\mathbf{X}(k)$  或  $\mathbf{Y}(k)$  所进行的扩张使体积超过了  $\theta$  则禁止扩张。如果已建各模糊区都禁止扩张，则建立一个新模糊区，其细节和几种修正算法将在下面详述。Simpson 网络的这种学习算法虽然也是在线自适应式的，但是每个已建模糊区的超立方体体积达到  $\theta$  后，就不能再随环境的缓慢变化而自适应变化。

### 5.4.2 Fuzzy ART 的学习算法<sup>[22,23,26,27]</sup>

学习分成结构学习、参数学习和后处理等三个阶段。结构学习阶段按在线自适应方式对输入向量和输出向量空间进行模糊划分并且建立二者之间的联系。参数学习阶段用 BP 算法对各隶属函数的参数（超立方体各维的最小和最大点）进行精细调节。后处理阶段用于去除一些冗余的规则。

#### 1. Fuzzy ART 的结构学习

设按节拍  $k = 0, 1, \dots$  输入训练样本对  $(\mathbf{X}(k), \mathbf{Y}(k))$  其中  $\mathbf{X}$  是一个  $N$  维输入行向量， $\mathbf{Y}$  是一个  $M$  维理想输出行向量。在刚开始时，输入和输出空间皆未作任何划分也未建立任何规则。下面分别讨论输入空间划分、输出空间划分和规则建立的算法。

##### (1) 输入空间划分的算法

###### 预处理

设  $\mathbf{X}$  的每一个分量  $x_i \in [0, 1], i = 1 \sim N$ 。如果不满足这一条件 则应通过线性变换使之满足（见第 2 章 2.5.7 小节的 1）。

###### 扩张

将  $\mathbf{X} = [x_1, x_2, \dots, x_N]$  扩张为一个  $2N$  维向量  $\mathbf{X}^c = [x_1, x_2, \dots, x_N, 1 - x_1, 1 - x_2, \dots, 1 - x_N]$  它的后  $N$  个分量是前  $N$  个分量的互补值 即  $x_{i+N} = 1 - x_i, i = 1 \sim N$ 。将  $\mathbf{X}^c$  作为学习阶段网络的输入。

###### 选择

设节拍  $k$  时输入向量空间中已建立了  $q(k)$  个模糊区间（注意： $q(0) = 0$ ）每个模糊空间用它的  $N$  维超立方体的最小点及最大点  $U_n^i, V_n^i, n = 1 \sim N$  来描述。它们构成一个  $2N$  维向量  $\mathbf{W}^i = [U_1^i, U_2^i, \dots, U_N^i, 1 - V_1^i, 1 - V_2^i, \dots, 1 - V_N^i], i = 1 \sim q(k)$ 。注意 这个向量的前  $N$  个分量是超立方体的  $N$  个最小点 后  $N$  个分量是  $N$  个最大点的补值。这样，可以用下式计算输入  $\mathbf{X}^c$  相对于各模糊区间  $\mathbf{W}^i$  的选择度  $t_i$

$$t_i = \frac{|\mathbf{X}^c \wedge \mathbf{W}^i|}{\alpha + |\mathbf{W}^i|}, \quad i = 1 \sim q(k) \quad (5-78)$$

符号“ $\wedge$ ”表示模糊交 (conjunction) 若有  $\mathbf{X} = [x_1, \dots, x_N], \mathbf{Y} = [y_1, \dots, y_N], \mathbf{X} \wedge \mathbf{Y} = \mathbf{Z} = [z_1, \dots, z_N]$  则  $z_i = \min\{x_i, y_i\}, i = 1 \sim N$ 。  $|\cdot|$  表示向量模，即  $|\mathbf{X}| = \sum |X_n|$ 。 $\alpha$  是一个略大于 0 的常数，其作用是防止  $|\mathbf{W}^i|$  值过小时  $t_i$  值过大 从而造成溢出 实际上这种情况几乎不会发生，所以实际操作时可以令  $\alpha \approx 0$ 。 $t_i$  指示了  $\mathbf{X}^c$  对  $\mathbf{W}^i$  的隶属程度。设

$t_j$  是诸  $t_i$  中最大者 即

$$t_j = \max\{t_i\} \quad (5-79)$$

匹 配

验证  $\mathbf{X}^c$  与  $\mathbf{W}^j$  的匹配程度, 为此计算下式是否成立

$$\frac{|\mathbf{X}^c \wedge \mathbf{W}^j|}{|\mathbf{X}^c|} \geq \rho \quad (5-80)$$

其中  $0 < \rho < 1$ ,  $\rho$  称为警戒参数。如果此式成立则称进入了谐振状态, 这时用下列 ⑥ 项算法对  $\mathbf{W}^j$  进行自适应调整。如果此式不满足 则舍弃  $\mathbf{W}^j$  转而验证与  $t_i$  中第二大者相应的  $\mathbf{W}^i$  和  $\mathbf{X}^c$  之间的匹配度, 如能满足式 (5-80) 则可用 ⑥ 项算法对其调整; 反之则验证第三大者。依此类推 若终于找到了 即转向 ⑥ 并且令  $q(k+1) = q(k)$ 。若始终找不到 则用下列 ⑤ 的方法建立一个新模糊区, 其编号是  $q(k) + 1$ 。这时令  $q(k+1) = q(k) + 1$ 。

建立一个新模糊区  $\mathbf{W}^{q(k)+1}$ ,

$$\mathbf{W}^{q(k)+1} = \mathbf{X}^c \quad (5-81)$$

新建模糊区的超立方体是  $N$  维空间中的一个点, 此点即是  $\mathbf{X}$ , 即  $U_n^{q(k)+1} = V_n^{q(k)+1} = x_n$ ,  $n = 1 \sim N$ 。

⑥ 自适应调整

若  $\mathbf{X}^c$  与  $\mathbf{W}^j$  的匹配程度满足式(5-80) 则对  $\mathbf{W}^j$  进行自适应调整。设调整前的  $\mathbf{W}^j$  用  $\mathbf{W}_{old}^j$  表示 调整后用  $\mathbf{W}_{new}^j$  表示 则

$$\mathbf{W}_{new}^j = \eta(\mathbf{X}^c \wedge \mathbf{W}_{old}^j) + (1 - \eta) \mathbf{W}_{old}^j \quad (5-82)$$

其中  $(\mathbf{X}^c \wedge \mathbf{W}_{old}^j)$  是一个在  $\mathbf{W}_{old}^j$  基础上扩大的超立方体。其第  $n$  维的最小点  $U_{n\ new}^j = \min(x_n, U_{n\ old}^j)$  最大点  $V_{n\ new}^j = \max(x_n, V_{n\ old}^j)$ 。  $0 \leq \eta \leq 1$ ,  $\eta$  用来控制学习的速度,  $\eta$  越接近 1 则学得越快。

这个算法的两个控制参数是  $\rho$  和  $\eta$ 。若  $\rho$  接近于 1, 则模糊空间划分得越细, 相应的模糊区间数越多。 $\eta$  则控制记忆的弹性和牢固度。若  $\eta$  接近于 1, 则系统对环境的变化适应越快, 即弹性越好。反之, 则记忆的牢固度越好。

应指出,  $\mathbf{X}^c$  这个扩张向量只有在对网络进行训练时才有用, 而在正常工作时只需用未扩张的  $\mathbf{X}$ 。

(2) 输出空间划分的算法

此算法与前述输入空间划分算法完全相同。首先通过预处理和扩张将  $M$  维向量  $\mathbf{Y}$  转换为  $2M$  维向量  $\mathbf{Y}^c$  然后按照同样的原则 以节拍  $k$  逐次建立  $r(k)$  个模糊区 ( $r(0) = 0$ )。每个模糊区用它的  $M$  维超立方体最小点  $U_m^l$  和最大点补值  $(1 - V_m^l)$  构成的  $2M$  维向量  $\mathbf{W}^l = [U_1^l, U_2^l, \dots, U_M^l, 1 - V_1^l, 1 - V_2^l, \dots, 1 - V_M^l]$  来描述  $l = 1 \sim r(k)$ 。

应该指出, 在实现函数逼近时  $\mathbf{Y}$  是一个  $\mathbf{R}^M$  中的向量且每一分量取值区间为  $[0, 1]$ 。而对于分类问题, 若共有  $M$  种类别, 则无须进行输出空间划分, 只需令系统的模糊推理规则数  $r = M$  即可。对于聚类问题, 聚类个数等于  $r$  它可以等于  $q(k)$  (即每个输入空间的模糊区间相应于一个聚类), 也可以小于  $q(k)$  (这时若干个输入模糊区间合并为一个聚类)。

### (3) 推理规则建立

带时间节拍  $k$  的规则数依不同应用而异。对于函数逼近问题，每一个输出模糊区间相应于一条模糊推理规则的结论，所以规则数即等于  $r(k)$ 。对于分类问题，设分成  $M$  类，则规则数永远等于  $M$ 。对于聚类问题，可以先令规则数等于  $q(k)$ （一个输入模糊区间一条规则），然后通过后处理予以合并。下面讨论如何建立规则。这就是如何建立  $q(k)$  个前提和  $r(k)$  个结论之间的联系（对于分类问题， $r(k) \equiv M$ ）。下面分几种情况来讨论。

若在节拍  $k$ ，输入空间建立一个新模糊区  $W^{q(k)+1}$ ，那么不管输出空间是建立了一个新模糊区  $W^{r(k)+1}$  还是未建立新模糊区（对于后者，设  $W^L$  是输出空间各模糊区选择度最大且满足匹配条件者），则在  $W^{q(k)+1}$  与  $W^{r(k)+1}$  或  $W^L$  之间建立模糊推理关系。若在节拍  $k$  输出空间建立了一个新模糊区  $W^{r(k)+1}$ （或在分类时启用了—个未用过的分类端），那么不管输入空间新建—  $W^{q(k)+1}$  或未建新区，只要  $W^J$  是选择度最大且满足匹配条件者，则在  $W^J$  或  $W^{q(k)+1}$  与  $W^{r(k)+1}$  之间建立推理关系。

如果输入和输出空间皆未建立新模糊区，二者的选择度最大且满足匹配条件者分别是  $W^J$  和  $W^L$ 。当  $W^L$  和  $W^J$  之间原已建立推理关系，则无须更改。当  $W^L$  和  $W^J$  之间原先未建立推理关系，则应废除  $W^J$  并在选择度较小且满足匹配条件的各  $W^i$  中搜寻一个与  $W^L$  原已建立推理关系的模糊区。如果找到了，则以该模糊区作为自适应调整的对象；如果找不到，则新建—模糊区。这时可以按 ① 所述的原则建立推理关系。

当结构学习结束时，设网络的输入空间划分为  $q$  个模糊区，其隶属函数为（见式 5-16）

$$\mu_A^i(\mathbf{X}) = \prod_{n=1}^N \mu_{A_n}^i(x_n), \quad i = 1 \sim q$$

其中各  $\mu_{A_n}^i(x_n)$  按式 5-18) 计算，每个  $\mu_{A_n}^i(x_n)$  用  $U_n^i, V_n^i, \gamma_n^i = \gamma$  来描述， $\gamma$  一般取为一个预定的常数（例如  $\gamma = 5$ ）， $U_n^i, V_n^i$  在结构学习中已确定。诸  $\mu_{A_n}^i(x_n)$  由网络的第 2 层完成其计算，诸  $\mu_A^i(\mathbf{X})$  的计算由第 3 层完成。网络的第 4 层完成  $r$  条推理计算（见式 5-51），得到

$$\mu^l(k) = \mu^l(\mathbf{X}(k)) = \mu_A^{i^*(l)}(\mathbf{X}(k)), \quad l = 1 \sim r$$

$\mu^l(k)$  即是输入为  $\mathbf{X}(k)$  时，第  $l$  条规则成立的隶属度。对于函数逼近问题，每条规则的结论与一个输出模糊空间对应，其隶属函数为（可仿照式 5-16）写出）

$$\mu_B^l(\mathbf{Y}) = \prod_{m=1}^M \mu_{B_m}^l(y_m), \quad l = 1 \sim r$$

其中每个  $\mu_{B_m}^l(y_m)$  与式 5-18 形式类似，用  $U_m^l, V_m^l, \gamma$  描述。诸  $\mu_{B_m}^l(y_m)$  及  $\mu_B^l(\mathbf{Y})$  在网络处于工作（运行）时，实际上无须进行计算。实际用得着的是存储在网络中的各个参数  $U_m^l$  和  $V_m^l, m = 1 \sim M, l = 1 \sim r$ 。按照式 5-77) 可得

$$y_m = \frac{\frac{1}{2} \sum_{l=1}^r (U_m^l + V_m^l) \mu^l(\mathbf{X})}{\sum_{l=1}^r \mu^l(\mathbf{X})}, \quad m = 1 \sim M \quad (5-83)$$

这样，在参数学习时需要进一步细调的参数是全部  $U_n^i, V_n^i$  和  $U_m^l, V_m^l$ 。

对于分类问题，每一条规则  $l$  的结论与一个类别对应，这时类别数  $M$  即等于规则数  $r$ ；若  $l = m$  则  $U_m^l = V_m^l = 1$ ，若  $l \neq m$  则  $U_m^l = V_m^l = 0$ 。这时得到（注意： $M = r$ ）

$$y_m = \frac{\mu^m(\mathbf{X})}{\sum_{l=1}^M \mu^l(\mathbf{X})}, \quad m = 1 \sim M \quad (5-84)$$

这时只有  $U_n^i, V_n^i$  需要在参数学习进行调节。

## 2. Fuzzy ART 的参数学习

函数逼近问题和分类问题的参数学习算法相同，只是后者的各输出层参数是固定的，无须进行学习。所以只需讨论前者就够了。

参数学习用 BP 算法 (随机梯度法) 设节拍  $k$  的训练样本是  $(\mathbf{X}(k), \hat{\mathbf{Y}}(k))$  实际输出是  $\mathbf{Y}(k)$  (为简化起见，在下面的推导中有时将变量  $k$  略而不书)。学习目的是求上述各参数的最佳值，使得下列目标函数  $E$  达到最小：

$$E = \|\hat{\mathbf{Y}} - \mathbf{Y}\|^2 = \sum_{m=1}^M (\hat{y}_m - y_m)^2 \quad (5-85)$$

其中  $y_m$  用式 (5-83) 计算。按照随机梯度法，首先可用下列迭代计算求输出层的最佳系数，

$$\left. \begin{aligned} U_m^l(k+1) &= U_m^l(k) - \alpha \frac{\partial E}{\partial U_m^l}, \quad l = 1 \sim r \\ V_m^l(k+1) &= V_m^l(k) - \alpha \frac{\partial E}{\partial V_m^l}, \quad m = 1 \sim M \end{aligned} \right\} \quad (5-86)$$

其中

$$\frac{\partial E}{\partial U_m^l} = \frac{\partial E}{\partial V_m^l} = -(\hat{y}_m - y_m) \frac{\mu^l(\mathbf{X})}{\sum_{l=1}^r \mu^l(\mathbf{X})}$$

其次，为计算输入空间各隶属函数的最佳系数，可用下列迭代计算：

$$\left. \begin{aligned} U_n^i(k+1) &= U_n^i(k) - \alpha \frac{\partial E}{\partial U_n^i}, \\ V_n^i(k+1) &= V_n^i(k) - \alpha \frac{\partial E}{\partial V_n^i}, \quad i = 1 \sim q, \quad n = 1 \sim N \end{aligned} \right\} \quad (5-87)$$

$$\left. \begin{aligned} \frac{\partial E}{\partial U_n^i} &= -2 \sum_{m=1}^M (\hat{y}_m - y_m) \frac{\partial y_m}{\partial U_n^i}, \quad \frac{\partial y_m}{\partial U_n^i} = \sum_{l=1}^r \frac{\partial y_m}{\partial \mu^l(k)} \cdot \frac{\partial \mu^l(k)}{\partial U_n^i} \\ \frac{\partial E}{\partial V_n^i} &= -2 \sum_{m=1}^M (\hat{y}_m - y_m) \frac{\partial y_m}{\partial V_n^i}, \quad \frac{\partial y_m}{\partial V_n^i} = \sum_{l=1}^r \frac{\partial y_m}{\partial \mu^l(k)} \cdot \frac{\partial \mu^l(k)}{\partial V_n^i} \end{aligned} \right\}$$

其中  $\partial y_m / \partial \mu^l(k)$  可用式 (5-55) 计算 (可令该式中的各  $\sigma_0^l(k)$  和  $\sigma_0^s(k)$  皆等于 1)。再利用式 (5-51) 和式 (5-18) (令其中  $\gamma_n^i \equiv \gamma$ ) 可以求得

$$\left. \begin{aligned} \frac{\partial \mu^l(k)}{\partial U_n^i} &= \frac{\partial \mu_{\Lambda}^{i*^{(l)}}(\mathbf{X})}{\partial U_n^i} = \prod_{\substack{p=1 \\ p \neq n}}^N \mu_{\Lambda p}^{i*^{(l)}}(x_p) \cdot \frac{\partial \mu_{\Lambda n}^{i*^{(l)}}(x_n)}{\partial U_n^i}, \\ \frac{\partial \mu^l(k)}{\partial V_n^i} &= \frac{\partial \mu_{\Lambda}^{i*^{(l)}}(\mathbf{X})}{\partial V_n^i} = \prod_{\substack{p=1 \\ p \neq n}}^N \mu_{\Lambda p}^{i*^{(l)}}(x_p) \cdot \frac{\partial \mu_{\Lambda n}^{i*^{(l)}}(x_n)}{\partial V_n^i}, \end{aligned} \right\} \quad n = 1 \sim N \quad (5-88)$$

其中



$$\left. \begin{aligned} \frac{\partial \mu_{A_n}^{i^*(l)}(x_n)}{\partial U_n^i} &= \begin{cases} -\gamma, & 0 \leq (U_n^i - x_n)\gamma < 1 \text{ 且 } i = i^*(l) \\ 0, & \text{其他} \end{cases} \\ \frac{\partial \mu_{A_n}^{i^*(l)}(x_n)}{\partial V_n^i} &= \begin{cases} \gamma, & 0 \leq (x_n - V_n^i)\gamma < 1 \text{ 且 } i = i^*(l) \\ 0, & \text{其他} \end{cases} \end{aligned} \right\} \quad (5-89)$$

注意，取结构学习结束时获得的这些参数值作为上列迭代计算的初值。

### 3. 后处理

在结构和参数学习过程中，输入或输出空间中的各模糊区的超立方体有可能放大、缩小或迁移。如果在输入或输出空间中一个超立方体完全包含了另一个超立方体，这说明这两个模糊区是多余的。图 5-8所示是当输入空间维数  $N = 2$  时，一个超立方体  $f$  完全包含另一超立方体  $g$  的例子。由于二者的重叠造成冗余，下面介绍一种去除冗余的可能方案<sup>[26]</sup>。可以采取保留大立方体  $\mu_A^f(\mathbf{X})$  的方案，也可以采取保留小立方体  $\mu_A^g(\mathbf{X})$  的方案，而把另一个予以去除。如按后者实施，则在超立方体  $g$  以外而在  $f$  以内的训练样本应予以重新启用。对输出空间可照此办理。当某个输入空间中的超立方体被取消后，与之相联系的模糊推理关系随之取消。如果这造成了输出空间中某个超立方体完全没有了与之相应的输入空间超立方体，则应将保留下来的输入超立方体与其建立联系。当某个输出空间超立方体被取消后，与之已建立联系的输入超立方体都应保留下来的输出超立方体之间建立联系。保留大立方体而去除小立方体的方案使空间划分变粗。反之，则空间划分变细。

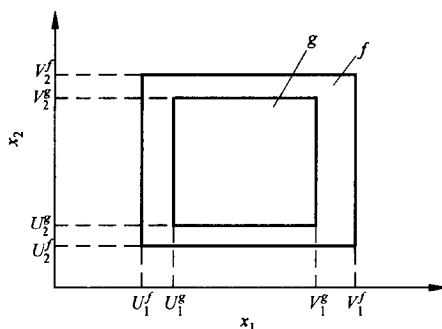


图 5-8 一个超立方体包含另一超立方体的例子

### 5.4.3 用于分类的 Simpson MIN-MAX 网络

此网络的结构与运作原理与一般的基于 Fuzzy ART 的用于分类的 5 层 FNN 相同，其输入空间分成  $q$  个模糊区间，每个区间用一个超立方体隶属函数描述（式（5-18）和式（5-16））。系统的规则数  $r$  与分类类别数  $M$  相等， $r$  条规则  $R^l, l = 1 \sim r (r = M)$ 。由输入向量  $\mathbf{X}$  可以求得对于  $M$  种类别的隶属度  $\mu^l(\mathbf{X}), l = 1 \sim M$ （见式（5-3）和式（5-51））。此网络也采用在线自适应的学习方式，每个节拍  $k$  输入一对训练向量  $(\mathbf{X}(k), \mathbf{Y}(k))$ ，其中  $\mathbf{Y}(k) = [\hat{y}_1(k), \dots, \hat{y}_M(k)]$  是指示  $\mathbf{X}(k)$  所属理想类别的向量，即当  $\mathbf{X}(k)$  属第  $C$  类时  $\hat{y}_C(k) = 1$ ； $\hat{y}_m(k) = 0$  当  $m \neq C$ 。但是学习方法与一般 Fuzzy ART 不同。下面介绍这一算法<sup>[24]</sup>

### 1. 预处理

$\mathbf{X}$  的每一维分量  $x_n$  应在  $[0,1]$  范围内。如不在此范围内，应通过线性变换使这一条件得到满足（见 5.4.2）。

### 2. 扩张

计算按节拍  $k = 0, 1, 2, \dots$  进行，下面具体讨论。

(1) 对于  $k = 0$ ，输入为  $\mathbf{X}(0)$  且  $\mathbf{X}(0)$  属于  $C(0)$  类。建立第 1 个输入空间超立方体  $\mu_A^1(\mathbf{X})$ ，它的最小点和最大点参数为  $U_n^1 = V_n^1 = x_n(0)$ 。令此立方体与输出中的第  $C(0)$  类建立推理关系。

(2) 对于  $k \geq 1$ ，输入为  $\mathbf{X}(k)$  且  $\mathbf{X}(k)$  属于第  $C(k)$  类。在已建立的各个输入空间的超立方体中搜索满足下列条件者：第一，已与第  $C(k)$  类建立了推理关系 第二 满足下列可扩张条件：

$$\sum_{n=1}^N [\max(V_n^i, x_n(k)) - \min(U_n^i, x_n(k))] \leq N\theta \quad (5-90)$$

$\theta$  是一个限制超立方体体积的参数， $\theta$  越小则立方体的体积越小，即输入空间分得越细。文献[24] 中针对其应用实例选了  $\theta = 0.0175, 0.15, 0.175$  等进行了模拟计算。若此条件得到满足 则此超立方体  $\mu_A^i(\mathbf{X})$  的各参数  $U_{n \text{ old}}^i$  和  $V_{n \text{ old}}^i$  按下式扩展为  $U_{n \text{ new}}^i$  和  $V_{n \text{ new}}^i$

$$\left. \begin{aligned} U_{n \text{ new}}^i &= \min[U_{n \text{ old}}^i, x_n(k)], \quad n = 1 \sim N \\ V_{n \text{ new}}^i &= \max[V_{n \text{ old}}^i, x_n(k)], \quad n = 1 \sim N \end{aligned} \right\} \quad (5-91)$$

如果这两个条件之一不能满足，则应建立一个新的输入空间模糊区，其第  $n$  维的最小点和最大点即等于  $x_n(k)$ 。令其与  $C(k)$  类建立推理关系。

### 3. 压缩

Simpson 网络规定，在分类应用中，如果输入空间中两个超立方体所建立的推理关系导向不同的类别时，这两个超立方体至少有一维是不交叠的。设其第  $\nu$  维不交叠，则下列二式中必有一个成立（设  $f$  和  $g$  是两个立方体）：

$$U_\nu^f \leq V_\nu^f \leq U_\nu^g \leq V_\nu^g \quad \text{或} \quad U_\nu^g \leq V_\nu^g \leq U_\nu^f \leq V_\nu^f$$

如果这两个立方体  $f$  和  $g$  所有各维都交叠，则应找到交叠最小的维，设其为  $\nu$  则交叠情况可表示为

$$U_\nu^f < U_\nu^g < V_\nu^f < V_\nu^g \quad \text{或} \quad U_\nu^g < U_\nu^f < V_\nu^g < V_\nu^f$$

且  $(V_\nu^f - U_\nu^g)$  或  $(V_\nu^g - U_\nu^f)$  较其他各维都小。找到后 可采用下列算法将标有“old”下标的老参数转变为标有“new”下标的新参数 从而消除交叠。如  $U_\nu^f \text{ old} < U_\nu^g \text{ old} < V_\nu^f \text{ old} < V_\nu^g \text{ old}$  则令

$$\begin{aligned} U_{\nu \text{ new}}^f &= U_{\nu \text{ old}}^f, & V_{\nu \text{ new}}^f &= \frac{1}{2}(U_{\nu \text{ old}}^g + V_{\nu \text{ old}}^f) \\ V_{\nu \text{ new}}^g &= V_{\nu \text{ old}}^g, & U_{\nu \text{ new}}^g &= \frac{1}{2}(U_{\nu \text{ old}}^g + V_{\nu \text{ old}}^f) \end{aligned}$$

这样 在 Simpson 网络的学习过程中，必须在每一节拍  $k$  检验一个扩张的超立方体与其他超立方体的交叠状况。如果两个立方体至少有一维不交叠或虽然交叠但连到同一个输出类别，就无须采取措施；反之，则应用上列算法将交叠消除（注意，只需将交叠最小维的交叠去除就够了。其他维仍允许存在交叠）。

在有些分类问题中，各不同类别所属的  $\mathbf{X}$  不是完全互斥的（非此即彼）。文献[28] 在

Simpson 网络的基础上略作修正，即可使网络用于这种分类问题。

#### 5.4.4 用于聚类的 Simpson MIN-MAX 网络

此网络用于将输入向量  $\mathbf{X}$  划归若干个聚类区。网络也按在线自适应的方式进行学习，按节拍  $k$  输入训练向量  $\mathbf{X}(k)$ ，每个聚类区相应于一个超立方体。开始时不存在任何聚类区，随着各  $\mathbf{X}(k)$  的依次输入，按上一节所述的预处理、扩张、压缩算法逐步建立各聚类区。此网络与前一节所述的分类网络的区别在以下两点。

第一 每个聚类区并不与  $\mathbf{X}$  所属的一种外界规定的类别相对应。因此，在进行扩张时，只需满足式 (5-90) 规定的条件即可，无须考虑其外在类别的推理关系。此外，它与一般聚类算法的区别在于，后者对于任一输入  $\mathbf{X}$  给出其隶属于某聚类区之值非 1 即 0 (完全隶属或完全不隶属) 而前者关于  $\mathbf{X}$  对于各聚类区的隶属值则在  $[0,1]$  区间中变化。

第二，对于前一节的分类网络，只要两个推理结果不同的超立方体之间有一维不交叠即无须进行压缩，否则只需去除交叠最小一维中的交叠即可 (其他各维无须变更)。而对于聚类网络，为了保证聚类的紧凑性，任意两个超立方体之间的每一维都不允许出现交叠。因此，在学习过程中每扩展一个超立方体时，都须检验其与其他超立方体是否存在交叠 (逐维检验并逐维予以消除。交叠有搭接和包容两种情况。设有  $f$  和  $g$  两个超立方体，其第  $n$  维的 MIN-MAX 点分别为  $(U_n^f, V_n^f)$  和  $(U_n^g, V_n^g)$  则搭接可表示为

$$U_n^f < U_n^g < V_n^f < V_n^g \quad \text{或} \quad U_n^g < U_n^f < V_n^g < V_n^f$$

这时可以用上一节所述的压缩方法将交叠去除。而包容可以表示为

$$U_n^f < U_n^g < V_n^g < V_n^f \quad \text{或} \quad U_n^g < U_n^f < V_n^f < V_n^g$$

现在以前一情况为例，说明消除这种交叠的方法。分成两种情况处理。

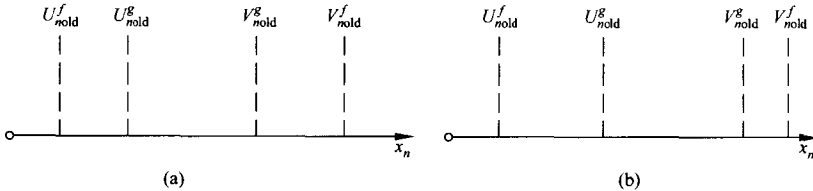


图 5-9 两种情况的  $U, V$

(1) 设压缩前的参数用  $(U_n^f, V_n^f)$  和  $(U_n^g, V_n^g)$  表示，压缩后的参数用  $(U_n^{f, \text{new}}, V_n^{f, \text{new}})$  和  $(U_n^{g, \text{new}}, V_n^{g, \text{new}})$  表示。第一种情况满足下列条件 如图 5-9(a) 所示：

$$V_n^f - U_n^g > V_n^g - U_n^f$$

这时压缩后的参数取下列值：

$$U_n^{f, \text{new}} = V_n^g, V_n^{f, \text{new}} = V_n^f, U_n^{g, \text{new}} = U_n^g, V_n^{g, \text{new}} = V_n^g$$

(2) 第二种情况的条件如下列，见图 5-9(b)：

$$V_n^f - U_n^g < V_n^g - U_n^f$$

这时按下式取得压缩后的参数值为

$$U_n^{f, \text{new}} = U_n^f, V_n^{f, \text{new}} = U_n^g, U_n^{g, \text{new}} = U_n^g, V_n^{g, \text{new}} = V_n^g$$

Simpson MIN-MAX 聚类网络的惟一控制参数是  $\theta$ 。当  $\theta$  取值较小时，可以获得较高的聚类精度，但是聚类区（超立方体）的个数相应地较多。此外，为了使每个聚类区稳定，训练集中的各个  $\mathbf{X}$  要重复使用多次，来对网络进行训练（即上述的算法中每一次扩张压缩为一轮，每一个  $\mathbf{X}$  要涉及多轮）文献[28]采用多分辨率的思路对 Simpson 聚类网络作了改进。通过与其他聚类算法比较后证明，原来的和修正的 Simpson 网络在很多种应用实例中都有较好的性能。

## 5.5 实现聚类的 FNN

### 5.5.1 HCM、FCM 和 KCN

如 2.4 节所述，设有训练集  $\mathbf{X}_p = [x_{p1}, \dots, x_{pN}]$ ,  $p = 1 \sim P$  聚类的目标是将  $\mathbf{X}_p$  所处的空间分为  $M$  个聚类区  $CL_i$ ,  $i = 1 \sim M$  每个区的质心是  $\mathbf{W}_i = [w_{i1}, \dots, w_{iN}]$ 。聚类区的划分依据是使某个目标函数  $J$  达到最小。聚类可用于分类（见 5.4 节）也可用于对输入向量编码（加标记）。后者用于数据的压缩和特征提取，可以进一步用于信号的传输、识别、存储等应用领域（如图像和语音的编码和识别）。 $J$  应该表征各种实际应用中需要达到的目标。现在采用的目标函数分成两种。第一种是一般聚类学习算法中采用的 HCM(hard c-means) 目标函数  $J$  (见式 (5-12))，为便于与其他算法比较， $J$  表示如下：

$$J = \sum_{i=1}^M \sum_{p=1}^P \mu_{pi} d_{pi}^2 \quad (5-92)$$

$$\mu_{pi} = \begin{cases} 1, & d_{pi} < d_{pj}, j = 1 \sim M, j \neq i \\ 0, & \text{其他} \end{cases} \quad (5-93)$$

文献[6]证明，当式 (5-93) 和下列式 (5-94) 得到满足时，

$$\mathbf{W}_i = \frac{\sum_{p=1}^P \mu_{pi} \mathbf{X}_p}{\sum_{p=1}^P \mu_{pi}}, \quad i = 1 \sim M \quad (5-94)$$

按式 (5-92) 定义的  $J$  将达到它的一个局部极小点（注意  $J$  是  $\mathbf{W}_i$ ,  $i = 1 \sim M$  的函数）这样，从随机设置的一组初值  $\mathbf{W}_i(0)$ ,  $i = 1 \sim M$  出发交替用式 (5-93) 和式 (5-94) 进行迭代计算 总能使  $J = J(\mathbf{W}_i, i = 1 \sim M)$  达到其某个局部最小点（在第 2 章中称此算法为 LBG 算法）

第二种是式 (5-13) 的 FCM 目标函数  $J_F$ ，

$$J_F = \sum_{i=1}^M \sum_{p=1}^P \mu_{pi}^m d_{pi}^2, \quad m \geq 1$$

文献[6]证明，当下列式 (5-95) 和式 (5-96) 所规定的条件得到满足时，

$$\mu_{pi} = \left[ \sum_{j=1}^M \left( \frac{d_{pi}^2}{d_{pj}^2} \right)^{\frac{1}{m-1}} \right]^{-1}, \quad i = 1 \sim M \quad (5-95)$$

$$\mathbf{W}_i = \frac{\sum_{p=1}^P \mu_{pi}^m \mathbf{X}_p}{\sum_{p=1}^P \mu_{pi}^m}, \quad i = 1 \sim M \quad (5-96)$$

$J_F = J_F(\mathbf{W}_i, i = 1 \sim M)$  将达到它的一个局部极小点。注意  $d_{pi}^2 = \|\mathbf{X}_p - \mathbf{W}_i\|^2$ ,  $d_{pj}^2 = \|\mathbf{X}_p - \mathbf{W}_j\|^2$  以及  $\mu_{pi}$  是  $\mathbf{X}_p$  对于以  $\mathbf{W}_i$  为质心的聚类区  $CL_i$  的模糊隶属度。输入空间中任一向量  $\mathbf{X}$  对于  $CL_i$  的隶属度  $\mu_A^i(\mathbf{X})$  可用下式计算：

$$\mu_A^i(\mathbf{X}) = \left[ \sum_{j=1}^M \left( \frac{\|\mathbf{X} - \mathbf{W}_i\|^2}{\|\mathbf{X} - \mathbf{W}_j\|^2} \right)^{\frac{1}{m-1}} \right]^{-1}, \quad i = 1 \sim M \quad (5-97)$$

事实上  $\mu_{pi} = \mu_A^i(\mathbf{X}_p)$ 。不难证明

$$0 \leq \mu_A^i(\mathbf{X}) \leq 1, \quad \forall i \quad (5-98)$$

和

$$\sum_{i=1}^M \mu_A^i(\mathbf{X}) = 1 \quad (5-99)$$

式(5-99)对于一般的模糊隶属函数无须满足，而对于模糊聚类隶属函数则必须满足。

式(5-95)和(5-97)中  $m$  的取值范围是  $[1, \infty)$ 。

当  $m = 1$ ,  $\mu_A^i(\mathbf{X})$  满足下式，

$$\mu_A^i(\mathbf{X}) = \begin{cases} 1, & \|\mathbf{X} - \mathbf{W}_i\|^2 < \|\mathbf{X} - \mathbf{W}_j\|^2, j = 1 \sim M, j \neq i \\ 0, & \text{其他} \end{cases}$$

这时  $\mu_{pi}$  只能取 0 或 1 值,  $J_F$  退化为  $J_0$ 。即 HCM 是 FCM 的一个特例 ( $m = 1$ )。

若  $m \rightarrow \infty$  则

$$\mu_A^i(\mathbf{X}) \rightarrow \frac{1}{M}, \quad i = 1 \sim M$$

这时  $\mathbf{X}$  对各聚类区的模糊隶属度值相等，这相应于完全模糊化。而前述  $m = 1$  的情况相应于完全非模糊化，即对各聚类区作确定划分。

下面给出 FCM 迭代学习算法，其目标是针对给定的训练集  $\mathbf{X}_p, p = 1 \sim P$  求一组最佳的聚类区质心  $\mathbf{W}_i^*, i = 1 \sim M$  使得  $J_F$  达到一个局部极小点 ( $m$  值给定)。此算法是 HCM 迭代学习算法的推广，可称为广义 LBG 算法。算法由交替迭代使用式(5-95)和式(5-96)构成。

(1) 给定  $M, m$  和  $\epsilon > 0$  ( $\epsilon$  是一个小正数)，给定训练集  $\mathbf{X}_p, p = 1 \sim P$ 。

(2) 设定随机初始质心  $\mathbf{W}_i(0) \in R^N, i = 1 \sim M$ 。设定最大迭代计算次数  $K_{\max}$ 。

(3) 令迭代节拍  $k = 1$ 。

(4) 计算

$$\mu_{pi}(k) = \left[ \sum_{j=1}^M \left( \frac{\|\mathbf{X}_p - \mathbf{W}_i(k-1)\|^2}{\|\mathbf{X}_p - \mathbf{W}_j(k-1)\|^2} \right)^{\frac{1}{m-1}} \right]^{-1}, \quad i = 1 \sim M, p = 1 \sim P$$

(5) 计算

$$\mathbf{W}_i(k) = \frac{\sum_{p=1}^P \mu_{pi}^m(k) \mathbf{X}_p}{\sum_{p=1}^P \mu_{pi}^m(k)}, \quad i = 1 \sim M$$

(6) 计算并比较

$$\sum_{i=1}^M \| \mathbf{W}_i(k) - \mathbf{W}_i(k-1) \|^2 < \epsilon?, \quad \forall i = 1 \sim M$$

若回答为是 转 (8) 若回答为否 转 (7)。

(7)  $k < K_{\max}$ ?

若回答为是 令  $k = k + 1$  转 (4) 若回答为否 转 (8)。

(8) 结束。输出  $\mathbf{W}_i(k), i = 1 \sim M$ 。

应该指出, 这是一个批处理算法。如果  $J_F = J_F(\mathbf{W}_i, i = 1 \sim M)$  有多个极小点 则此算法将依初值设置使  $\mathbf{W}_i$  收敛到某个局部极小点。

为了用神经网络实现模糊聚类, 很自然地首先应考虑一种效能很好的用于聚类的神经网络——KCN(Kohonen clustering network) 或称为 SOM。这种自组织神经网络已在第 3 章中讨论过。为了研究如何将 KCN 和 FCM 结合起来, 下面先给出 KCN 的一个标准算法<sup>[30]</sup>

(1) 给定  $M$  和  $\epsilon > 0$ ; 给定训练集  $\mathbf{X}_p, p = 1 \sim P$ 。

(2) 设定随机初始质心  $\mathbf{W}_i(0), i = 1 \sim M$  设定最大迭代计算次数  $K_{\max}$ 。

(3) 设定初始步幅系列  $\alpha_{j^*}(1), j^* = 1 \sim M$  (其取值范围为  $(0, 1)$  且当  $j^* = 1$  时其取值较大 当  $j^*$  增大时其取值逐渐下降) 给出  $\alpha_{j^*}(k), k = 2 \sim K_{\max}$ 。

(4) 设定初始邻域界限  $\psi(1), \psi(1)$  为不小于 1 的整数 (一般可令  $\psi(1) = M$ ) 给出  $\psi(k), k = 2 \sim K_{\max}$ 。

(5) 令迭代节拍  $k = 1$ 。

(6) 令  $\mathbf{W}_i^{(1)}(k-1) = \mathbf{W}_i(k-1), i = 1 \sim M$  令  $p = 1$ 。

(7) 计算  $\mathbf{X}_p$  与  $\mathbf{W}_i^{(p)}(k-1)$  的欧氏距离  $d_{ip}(k)$

$$d_{ip}^2(k) = \| \mathbf{X}_p - \mathbf{W}_i^{(p)}(k-1) \|^2, i = 1 \sim M$$

(8) 将各  $d_{ip}^2(k)$  按从小到大的次序排列如下:

$$d_{1^*p}^2(k) < d_{2^*p}^2(k) < \dots < d_{M^*p}^2(k)$$

(例如  $d_{1^*p}^2(k) < d_{2^*p}^2(k) < \dots < d_{2^*p}^2(k)$  则  $1^* = 3, 2^* = 7, \dots, M^* = 2$ 。  $1^*$  为相对于  $\mathbf{X}_p$  的优胜者编号  $2^*$  为第 2 名的编号, 等等)。

(9) 对于  $i = 1^*, 2^*, \dots, [\psi(k)]^*$  按下式进行迭代计算,

$$\mathbf{W}_{j^*}^{(p+1)}(k) = \mathbf{W}_{j^*}^{(p)}(k-1) + \alpha_{j^*}(k) [\mathbf{X}_p - \mathbf{W}_{j^*}^{(p)}(k-1)]$$

$$j^* = 1^* \sim 2^* \sim \dots \sim [\psi(k)]^*$$

(10)  $p < P$ ?

若回答为是 令  $p = p + 1$  转 (7) 若回答为否 令  $\mathbf{W}_i(k) = \mathbf{W}_i^{(p)}(k)$  转 (11)。

(11) 计算并比较

$$E(k) = \sum_{i=1}^M \| \mathbf{W}_i(k) - \mathbf{W}_i(k-1) \|^2 < \epsilon?$$

若回答为是, 转 (13) 若回答为否 转 (12)。

(12)  $k < K_{\max}$ ?

若回答为否, 转 (13) 若回答为是 令  $k = k + 1$  转 (6)。

(13) 结束 输出  $\mathbf{W}_i(k), i = 1 \sim M$ 。

说明：上述程序中  $\alpha_{j^*}(k)$  应随着  $k$  的增加而缓慢下降（例如  $\alpha_{j^*}(k) = \alpha_{j^*}(1)/k$ ）。 $\psi(k)$  也应随  $k$  的增加而缓慢下降，当  $k = K_{\max}$  时应使  $\psi(K_{\max}) = 1$ 。应该指出 KCN 学习算法是一种非批处理的序贯算法，具有随机性。最终学习结果不但与初值有关而且与训练集中诸向量的排序有关。学习算法中  $\alpha_{j^*}(k)$  和  $\psi(k)$  两个序列参数必须恰当选择。此算法并未确定一个目标函数并通过迭代计算使之达到极小，因此是一种启发式的算法。尽管如此，KCN 在各种实际领域中具有良好效果（见第 3 章）。这一节将讨论若干种 FCM 与 NN 包括 KCN）相结合的方案，其目的是发挥二者的优势。

## 5.5.2 FKCEN

### 1. 序贯式的 FKCEN<sup>[31]</sup>

此学习算法仍在上节 KCN 学习算法的框架中进行，只是将其中的步幅  $\alpha_{j^*}(k)$  改为按下式计算（参见式 5-97）：

$$\alpha_{j^*}(k) = \mu_{pj^*}(k) = \left[ \sum_{j=1}^M \left( \frac{\|X_p - W_j(k-1)\|^2}{\|X_p - W_{j^*}(k-1)\|^2} \right)^{\frac{1}{m-1}} \right]^{-1}$$

这一算法将 FCM 的模糊隶属度概念引到了 KCN 的步幅控制，由  $m$  调节模糊程度，从而使之有更明确的意义。当然，更合理的做法是按下式来计算步幅：

$$\alpha_{j^*}(k) = \mu_{pj^*}(k)\alpha(k)$$

其中  $\alpha(k)$  是一个随  $k$  增加而递减的函数，例如  $\alpha(k) = \alpha_0/k, \alpha_0 > 0$ 。

### 2. 批处理式 FKCEN<sup>[29]</sup>

此网络的学习算法是对 FCM 和 KCN 的结合作进一步的改进。主要将序贯式改为批处理式以及将隶属函数中的幂值  $m$  由固定不变修正为随  $k$  而改变的函数  $m(k)$ 。前者可以使输入序列的排序对输出结果的影响得以排除；后者以一种较 KCN 更合理的方法（即通过改变  $m$  来改变模糊度）来改变各权值调整的邻域范围，从而有可能使迭代计算结果收敛到质量高的局部极小点。算法程序如下列。

(1) 给定  $M$  和  $\epsilon > 0$ ；给定训练集  $X_p, p = 1 \sim P$ 。

(2) 设定随机初始质心  $W_i(0), i = 1 \sim M$  设定最大迭代计算次数  $K_{\max}$ 。

(3) 给定初始和终止幂值  $m(0)$  和  $m(K_{\max})$ ，( $m(0)$  取较大值，例如 10。 $m(K_{\max})$  按需要而定，如最后需形成 HCM 则取  $m(K_{\max}) = 1$  如最后需形成 FCM 则取  $m(K_{\max}) > 1$  其具体值视所需的模糊度而定)。 $m(k)$  可用下式计算：

$$m(k) = [m(0) + k\Delta m], \quad \Delta m = \frac{-m(0) + m(K_{\max})}{K_{\max}}$$

(4) 令迭代节拍  $k = 0$ 。

(5) 按下式进行迭代计算：

$$W_i(k+1) = W_i(k) + \frac{\sum_{p=1}^P \alpha_{pi}(k)[X_p - W_i(k)]}{\sum_{p=1}^P \alpha_{pi}(k)}, \quad i = 1 \sim M$$

其中

$$\alpha_{pi}(k) = [\mu_{pi}(k)]^{m(k)}$$

$$\mu_{pi}(k) = \left[ \sum_{j=1}^M \left( \frac{\| \mathbf{X}_p - \mathbf{W}_i(k) \|^2}{\| \mathbf{X}_p - \mathbf{W}_j(k) \|^2} \right)^{\frac{1}{m(k)-1}} \right]^{-1}$$

(6) 计算并比较

$$E(k) = \sum_{i=1}^M \| \mathbf{W}_i(k+1) - \mathbf{W}_i(k) \|^2 < \epsilon?$$

若回答为是, 转 8) 若回答为否 转( 7)。

(7)  $k < K_{\max}$ ?

若回答为否, 转 8) 若回答为是, 令  $k = k+1$  转(5)。

(8) 结束, 输出  $\mathbf{W}_i(k+1), i = 1 \sim M$ 。

可以看到, 若  $m(k)$  等于不随  $k$  变化的某个常数  $m$  则此算法退化为 5.5.1 小节所述的 FCM 迭代学习算法。令  $m(k)$  在学习过程中由大变小, 正是吸取了 KCN 学习算法中改变调整邻域范围的经验, 从而使解的质量更高。此算法胜于 KCN 算法之处是, 去除了排序影响且赋予迭代计算中的步幅系数  $\alpha_{pi}(k)$  以明确的模糊隶属含意, 从而也能改善解的质量。

### 5.5.3 GCN(generalized clustering network)<sup>[32]</sup>

定义 GCN 的目标函数如下:

$$J_G = \sum_{i=1}^M \sum_{p=1}^P \mu_{pi} d_{pi}^2 \quad (5-100)$$

其中  $d_{pi}^2 = \| \mathbf{X}_p - \mathbf{W}_i \|^2$ 。  $\mathbf{X}_p, p = 1 \sim P$  是训练集  $\mathbf{W}_i, i = 1 \sim M$  是  $M$  个聚类区的质心。其中隶属度  $\mu_{pi}$  的定义如下。设  $i^*$  是输入为  $\mathbf{X}_p$  时的优胜者编号, 即满足下列条件:

$$d_{pi^*} < d_{pi}, \quad i = 1 \sim M, i \neq i^*$$

则可按下式求  $\mu_{pi}$

$$\mu_{pi} = \begin{cases} 1, & i = i^* \\ 1/D_p, & i = 1 \sim M, i \neq i^* \end{cases} \quad (5-101)$$

其中  $D_p = \sum_{j=1}^M d_{pj}^2$ 。可以看到 此  $\mu_{pi}$  定义不同于 FCM 中关于  $\mu_{pi}$  的定义(式 5-95))。  $J_G$  的定义也与  $J_F$  (式 5-13)) 不一致。下面介绍用随机梯度迭代计算求一组最佳的  $\mathbf{W}_i$  使  $J_G$  达到极小。这种算法称为 GLVQ 算法。

计算按迭代节拍  $k = 0, 1, 2, \dots, K_{\max}$  进行, 依次从训练集中取出一个向量  $\mathbf{X}(k)$  构成下列目标函数  $J_G(k)$  (参见式 5-101) 和式(5-100))。

$$J_G(k) = \sum_{i=1}^M \mu_i(k) d_i^2(k) \quad (5-102)$$

其中  $d_i^2(k) = \| \mathbf{X}(k) - \mathbf{W}_i(k) \|^2$ 。设  $i^*(k)$  是输入为  $\mathbf{X}(k)$  时的优胜者编号(为简化起见, 下文用  $i^*$  表示  $i^*(k)$ ) 则

$$\mu_i(k) = \begin{cases} 1, & i = i^* \\ \frac{1}{D(k)} = \frac{1}{\sum_{j=1}^M d_j^2(k)}, & i = 1 \sim M, i \neq i^* \end{cases} \quad (5-103)$$



计算从随机选择的初始质心  $\mathbf{W}_i(0), i = 1 \sim M$  , 出发, 按下列公式作迭代计算:

$$\mathbf{W}_i(k+1) = \mathbf{W}_i(k) + \Delta \mathbf{W}_i(k), \quad i = 1 \sim M$$

$$\Delta \mathbf{W}_i(k) = -\alpha(k) \nabla_{\mathbf{W}_i} J_G(k)$$

$\alpha(k)$  是一个随  $k$  增加而下降的步幅函数, 例如  $\alpha(k) = \alpha_0/k, \alpha_0 > 0$  。为了计算上式中的梯度  $J_G(k)$  可以改写为下列形式:

$$J_G(k) = \|\mathbf{X}(k) - \mathbf{W}_{i^*}(k)\|^2 + 1 - \frac{\|\mathbf{X}(k) - \mathbf{W}_{i^*}(k)\|^2}{D(k)}$$

其中  $D(k) = \sum_{j=1}^M \|\mathbf{X}(k) - \mathbf{W}_j(k)\|^2$ 。

这样, 可以求得

$$\nabla_{\mathbf{W}_i} J_G(k) = \begin{cases} -2[\mathbf{X}(k) - \mathbf{W}_i(k)] \left\{ \frac{D^2(k) - D(k) + \|\mathbf{X}(k) - \mathbf{W}_i(k)\|^2}{D^2(k)} \right\}, & i = i^* \\ -2[\mathbf{X}(k) - \mathbf{W}_i(k)] \frac{\|\mathbf{X}(k) - \mathbf{W}_{i^*}(k)\|^2}{D^2(k)}, & i = 1 \sim M, i \neq i^* \end{cases} \quad (5-104)$$

可以看到, 对于所有非获胜端, 其权的调整量要受到  $\|\mathbf{X}(k) - \mathbf{W}_{i^*}(k)\|^2$  加权。当  $\mathbf{X}(k) = \mathbf{W}_{i^*}(k)$  时, 所有非获胜端的权都不作调整,  $\mathbf{X}(k)$  偏离  $\mathbf{W}_{i^*}(k)$  越远, 它们的调整量越大, 而且各非获胜端受到同等的加权。获胜端权的调整量受其他非获胜端的影响表现于  $D(k)$ 。如果  $D(k) \gg 1$  则其影响可以略之不计。实验结果表明 这种算法的收敛结果对于初值的设置不敏感且计算效率较高<sup>[32]</sup>

但是 GLVQ 算法仍存在问题。文献[35]正确指出, 当  $D(k) \gg 1$  时 只有获胜端  $i^*$  能够得到调整而其他非获胜端都不能调整 (见式 (5-104)) 相反 当  $D(k) \ll 1$  时 式(5-104)中第 1 个等式的花括弧项将变成负值, 这时获胜端将反向 (错误方向) 调节且非获胜端将大幅度正向调节。这样, 前者退化为单点调节从而起不到改善初值影响的作用, 后者将使所得解完全错误。因此 只当  $D(k)$  不大不小时, GCN 算法效果较好。而  $D(k)$  的大小取决于输入向量  $\mathbf{X}$  的尺度和聚类的个数  $M$  这使得 GLVQ 算法受外在因素的影响太大。为此, 文献[34]提出了一种改进方案, 称为 GLVQ-F。分析发现 GLVQ 算法的种种问题皆出自目标函数  $J_G(k)$  (见式 5-102)) 中关于  $\mu_i(k)$  的定义 (见式 (5-103))。GLVQ-F 在保留 GLVQ 关于  $J_G(k)$  的定义并采用随机梯度法搜寻最优解的前提下, 将  $\mu_i(k)$  的定义改为用下式计算:

$$\mu_i(k) = \left[ \sum_{j=1}^M \left( \frac{\|\mathbf{X}(k) - \mathbf{W}_i(k)\|^2}{\|\mathbf{X}(k) - \mathbf{W}_j(k)\|^2} \right)^{\frac{1}{m-1}} \right]^{-1}, \quad i = 1 \sim M \quad (5-105)$$

这样可得到下列迭代计算公式:

$$\mathbf{W}_i(k+1) = \mathbf{W}_i(k) + \Delta \mathbf{W}_i(k), \Delta \mathbf{W}_i(k) = -\alpha(k) \nabla_{\mathbf{W}_i} J_G(k), \quad i = 1 \sim M$$

其中

$$\nabla_{\mathbf{w}_i} J_G(k) = \left( \frac{-2}{m-1} \right) [\mathbf{X}(k) - \mathbf{W}_i(k)] \left\{ (m-2)\mu_i(k) + \left[ \sum_{j=1}^M \left( \frac{\|\mathbf{X}(k) - \mathbf{W}_i(k)\|^2}{\|\mathbf{X}(k) - \mathbf{W}_j(k)\|^2} \right)^{\frac{2-m}{m-1}} \right] \mu_i^2(k) \right\} \quad (5-106)$$

此式的推导可在文献[34]中找到。特别地 当  $m = 2$  时，此式可简化为下列形式：

$$\nabla_{\mathbf{w}_i} J_G(k) = -2M\mu_i^2(k) [\mathbf{X}(k) - \mathbf{W}_i(k)]$$

此外，按照一般随机梯度算法准则，可以取步幅函数为  $\alpha(k) = \alpha_0/k, 0 < \alpha_0 < 1$  若规定最大迭代计算次数为  $K_{\max}$  也可取步幅函数为  $\alpha(k) = \alpha_0(1 - k/K_{\max}), 0 < \alpha_0 < 1$ 。

在 5.5.1 小节中已给出： $0 \leq \mu_i(k) \leq 1$  而且  $\sum_{i=1}^M \mu_i(k) = 1$  (见式 (5-98) 和式 (5-99))。

如果取  $m = 2$  则  $\mu_i(k) \propto (\|\mathbf{X}(k) - \mathbf{W}_i(k)\|^2)^{-1}$ 。GLVQ-F 在保持了 GLVQ 优点的同时 克服了它的缺点。与 FCM 相比，二者皆采用相同的隶属函数定义和目标函数的定义；二者的区别是：FCM 采用批处理算法而 GLVQ-F 采用随机梯度算法。与 FKCEN 相比，二者皆采用相同的隶属函数定义并使用序贯式的算法；二者的区别是：FKCEN 是一种启发式算法 它依据  $\mu_i(k)$  值的大小来控制每一个权值的调整量，而 GLVQ-F 具有明确的目标函数，调整量取决于目标函数的梯度值，这使序贯式的算法成为一种随机梯度算法。

#### 5.5.4 FALVQ

针对 GLVQ 所存在的缺点而进行改进的另一种算法是 FALVQ<sup>[34]</sup>。与 GLVQ 一致之处是它也采取随机梯度算法且具有相同形式的目标函数（见式 (5-102)）；二者的区别在于对非获胜端隶属函数的定义不同。

##### 1. FALVQ 的定义

按节拍  $k$  从训练集  $\mathbf{X}_s, p = 1 \sim P$  中依次取出向量  $\mathbf{X}(k)$  目标函数  $J_{FA}(k)$  定义如下：

$$J_{FA}(k) = \sum_{i=1}^M \mu_i(k) \|\mathbf{X}(k) - \mathbf{W}_i(k)\|^2 \quad (5-107)$$

设  $i^*(k)$  为输入  $\mathbf{X}(k)$  时的获胜端编号 (为简化, 下文中  $i^*(k)$  用  $i^*$  表示) 则  $\mathbf{X}(k)$  对于第  $i$  端的隶属函数  $\mu_i(k)$  定义如下：

$$\mu_i(k) = \begin{cases} 1, & i = i^* \\ \varphi \left( \frac{\|\mathbf{X}(k) - \mathbf{W}_{i^*}(k)\|^2}{\|\mathbf{X}(k) - \mathbf{W}_i(k)\|^2} \right), & i = 1 \sim M, i \neq i^* \end{cases} \quad (5-108)$$

其中函数  $\varphi(\cdot)$  取何种形式将留待本小节第 (3) 段中讨论。在这定义中,  $\mu_i(k)$  只取决于  $\mathbf{W}_{i^*}(k)$  和  $\mathbf{W}_i(k)$  而与其他  $\mathbf{W}_j(k)$  无关 从而使其与 FCM、FKCEN 和 GLVQ 的  $\mu_i(k)$  定义与其他  $\mathbf{W}_j(k)$  有关的情形不同 经过适当改写 式 (5-107) 可表示如下 (将式 (5-108) 代入)：

$$J_{FA}(k) = \|\mathbf{X}(k) - \mathbf{W}_{i^*}(k)\|^2 \left[ 1 + \sum_{\substack{i=1 \\ i \neq i^*}}^M \varphi \left( \frac{\|\mathbf{X}(k) - \mathbf{W}_{i^*}(k)\|^2}{\|\mathbf{X}(k) - \mathbf{W}_i(k)\|^2} \right) \times \frac{\|\mathbf{X}(k) - \mathbf{W}_i(k)\|^2}{\|\mathbf{X}(k) - \mathbf{W}_{i^*}(k)\|^2} \right]$$

$$= \| \mathbf{X}(k) - \mathbf{W}_{i^*}(k) \|^2 \left[ 1 + \sum_{\substack{i=1 \\ i \neq i^*}}^M \phi(z_i(k)) \right]$$

其中

$$z_i(k) = \frac{\| \mathbf{X}(k) - \mathbf{W}_{i^*}(k) \|^2}{\| \mathbf{X}(k) - \mathbf{W}_i(k) \|^2}$$

$$\phi(z_i(k)) = \phi(z_i(k))/z_i(k), \phi(z_i(k)) = z_i(k)\phi(z_i(k))$$

因  $i^*$  是获胜端 所以  $\| \mathbf{X}(k) - \mathbf{W}_i(k) \|^2 \geq \| \mathbf{X}(k) - \mathbf{W}_{i^*}(k) \|^2, \forall i \neq i^*$ , 即  $0 \leq z_i(k) \leq 1$ 。  $z_i(k) \rightarrow 0$  表示  $\mathbf{X}(k) \rightarrow \mathbf{W}_{i^*}(k)$  和 / 或  $\mathbf{X}(k)$  与  $\mathbf{W}_i(k)$  之间的欧氏距离趋于无穷大。  $z_i(k) \rightarrow 1$  则表示  $\mathbf{X}(k)$  与  $\mathbf{W}_{i^*}(k)$  和  $\mathbf{W}_i(k)$  二者之间的欧氏距离趋向一致。按照隶属函数的概念, 对于前者应有  $\phi(z_i(k)) \rightarrow 0$  对于后者则应有  $\phi(z_i(k)) \rightarrow \zeta_0$  ( $\zeta_0$  是一个略小于 1 或等于 1 的常数), 因此也可知 对于前者  $\phi(z_i(k)) < \infty$  应成立, 对于后者  $\phi(z_i(k)) \rightarrow \zeta_0$  应成立。

## 2. FALVQ 的学习算法

学习目的是从随机设置的初始权向量  $\mathbf{W}_i(0), i = 1 \sim M$  出发, 通过迭代计算求一组最佳权向量, 使目标函数  $\sum_{p=1}^P \sum_{i=1}^M \mu_i(\mathbf{X}_p) \| \mathbf{X}_p - \mathbf{W}_i \|^2$  达到最小。其中  $\mu_i(\mathbf{X}_p)$  可用式 (5-108) 计算 只需将式左侧括弧中的  $k$  换成  $\mathbf{X}_p$  式右侧的  $\mathbf{X}(k)$  换成  $\mathbf{X}_p$  即可) 迭代计算公式如下列:

$$\begin{aligned} \mathbf{W}_i(k+1) &= \mathbf{W}_i(k) + \Delta \mathbf{W}_i(k) \quad i = 1 \sim M \\ \Delta \mathbf{W}_i(k) &= -\alpha(k) \nabla_{\mathbf{W}_i} J_{\text{FA}}(k) \\ \nabla_{\mathbf{W}_i} J_{\text{FA}}(k) &= \begin{cases} -[\mathbf{X}(k) - \mathbf{W}_i(k)] \left[ 1 + \sum_{\substack{i=1 \\ i \neq i^*}}^M \omega_i(k) \right], & i = i^* \\ -[\mathbf{X}(k) - \mathbf{W}_i(k)] \nu_i(k), & i = 1 \sim M, i \neq i^* \end{cases} \end{aligned} \quad (5-109)$$

其中

$$\omega_i(k) = \left. \frac{d\phi(z)}{dz} \right|_{z=z_i(k)} = \phi(z_i(k)) + z_i(k) \left. \frac{d\phi(z)}{dz} \right|_{z=z_i(k)} \quad (5-110)$$

$$\nu_i(k) = \phi(z_i(k)) - z_i(k) \cdot \omega_i(k) \quad (5-111)$$

以上诸式的推导见文献 [34]。

## 3. $\phi(z)$ 和 $\psi(z)$ 的选择

### (1) HCM

令  $\phi(z) \equiv C_0 \in (0, 1], \phi(z) = C_0 z$  则可求得

$$\omega_i(k) = C_0, \nu_i(k) = 0, i = 1 \sim M, i \neq i^*$$

$$\Delta \mathbf{W}_i(k) = \begin{cases} \alpha(k) [1 + (M-1)C_0] [\mathbf{X}(k) - \mathbf{W}_i(k)], & i = i^* \\ 0, & i = 1 \sim M, i \neq i^* \end{cases}$$

### (2) FALVQ-1

令  $\phi(z) = 1/(1+z), \phi(z) = z/(1+z)$  则可求得:  $\omega_i(k) = 1/(1+z_i(k))^2, \nu_i(k) = z_i^2(k)/(1+z_i^2(k))^2, i = 1 \sim M, i \neq i^*$ 。可以看到 当  $z_i(k) \rightarrow 0$  时,  $\omega_i(k) \rightarrow 1$  且  $\nu_i(k) \rightarrow 0$ , 这时获胜端  $i^*$  得到调整而非获胜端  $i \neq i^*$  得不到调整 当  $z_i(k) \rightarrow 1$  时,  $\omega_i(k) \rightarrow 1/4$  且  $\nu_i(k) \rightarrow 1/4$ , 这时二者都得到机会调整, 但获胜端的调整量远大于非获胜端。

### (3) FALVQ-1'

令  $\psi(z) = 1/(1 + \beta z)$ ,  $\varphi(z) = z/(1 + \beta z)$ ,  $\beta > 0$ , 则可求得:  $\omega_i(k) = [1 - \beta z_i(k)/(1 + \beta z_i(k))]^2$ ,  $\nu_i(k) = \beta z_i^2(k)/(1 + \beta z_i(k))^2$ ,  $i = 1 \sim M, i \neq i^*$

### (4) FALVQ-2

令  $\psi(z) = e^{-z}$ ,  $\varphi(z) = ze^{-z}$  则可求得:  $\omega_i(k) = (1 - z_i(k))e^{-z_i(k)}$ ,  $\nu_i(k) = z_i^2(k)e^{-z_i(k)}$ ,  $i = 1 \sim M, i \neq i^*$ 。可以看到 当  $z_i(k) \rightarrow 0$  则  $\omega_i(k) \rightarrow 1$  且  $\nu_i(k) \rightarrow 0$  这时获胜端调整而非获胜端不调 当  $z_i(k) \rightarrow 1$  时  $\omega_i(k) \rightarrow 0$  且  $\nu_i(k) \rightarrow e^{-1}$ , 这时二者都调整且调整量比较接近。

### (5) FALVQ-2'

令  $\psi(z) = e^{-\beta z}$ ,  $\varphi(z) = ze^{-\beta z}$ , 则可求得:  $\omega_i(k) = (1 - \beta z_i(k))e^{-\beta z_i(k)}$ ,  $\nu_i(k) = \beta z_i^2(k)e^{-\beta z_i(k)}$ ,  $i = 1 \sim M, i \neq i^*$ 。

### (6) FALVQ-3

令  $\psi(z) = 1 - \gamma z$ ,  $\varphi(z) = z(1 - \gamma z)$ ,  $0 < \gamma \leq 1$ , 则可求得:  $\omega_i(k) = 1 - 2\gamma z_i(k)$ ,  $\nu_i(k) = \gamma z_i^2(k)$ ,  $i = 1 \sim M, i \neq i^*$ 。例如, 选  $\gamma = 0.5$  则有  $\omega_i(k) = 1 - z_i(k)$ ,  $\nu_i(k) = 0.5 z_i^2(k)$ 。当  $z_i(k) \rightarrow 0$  则  $\omega_i(k) \rightarrow 1$ ,  $\nu_i(k) \rightarrow 0$ , 这时获胜端调整而非获胜端不调; 当  $z_i(k) \rightarrow 1$  则  $\omega_i(k) \rightarrow 0$  且  $\nu_i(k) \rightarrow 0.5$  这时二者都调且调整量接近。

对于各种具体应用, 上列各种方案的效果及参数的选择在文献[34]中有详细实验结果。如规定最大迭代次数为  $K_{\max}$  则步幅函数可选为  $\alpha(k) = \alpha_0(1 - k/K_{\max})$ ,  $\alpha_0 > 0$ 。

## 5.6 FNN 在非线性动力系统辨识与控制中的应用

有关人工神经网络(主要是 MLFN)在非线性动力系统辨识、控制等领域中的应用问题已在 2.14 节中详细研究, 它们最后归结为一个函数逼近器的问题。由于 FNN 在实现函数逼近时有其独特优势, 将其应用于这些领域是很自然的事, 文献[12]给出了有关这一研究方向的近 20 年来发表的 58 篇论文索引, 可资参考。这一节将给出一些应用实例的介绍 特别要讨论 GMDH (group method of data handling) 技术在 FNN 中的应用。还要介绍一种特殊的控制问题——增强(reinforcement)学习问题。

### 5.6.1 GMDH 技术

在解决一个非线性动力系统辨识问题或控制问题时, 除了完成一般的函数逼近任务以外, 还需要针对具体的应用目标来选择 FNN 的输入变量和结构。就输入变量而言 如果有多个变量作为候选者, 就应从中选出若干最关键的变量构成网络输入向量。问题是应选多少个 选哪几个 按什么标准进行选择 对于网络结构而言 也存在着输入和输出空间各应划分为多少个模糊空间以及建立多少条模糊推理规则的问题。这里的根本问题是以什么标准来衡量各种可能的选择。GMDH 采用非偏准则 UC(unbiasedness criterion)来衡量各种选择的优劣<sup>[37]</sup>, 是一种著名的复杂非线性系统辨识方法。其优点是, 按此准则做出的网络变量和结构选择在对真实系统建模时具有优良的推广性能和鲁棒(robust)性能, 鲁棒性是指当训练数据中存在噪声时按此准则建立的网络对其最不敏感。GMDH 除了 UC

准则以外，另一项重要考虑因素是 FNN 的复杂性。输入变量选得越多，输入和输出空间划分得越细且规则数越多，当然会使函数逼近效果越好。但是，这会使计算开销增大且当训练数据不足时推广性能变差。下面介绍 GMDH。

设网络需完成由  $\mathbf{X}$  至  $y$  的映射。首先将训练集分成  $A, B$  两部分  $A: \{\mathbf{X}_{Ai}, \hat{y}_{Ai}\}, i = 1 \sim M_A; B: \{\mathbf{X}_{Bi}, \hat{y}_{Bi}\}, i = 1 \sim M_B$ 。其中  $\hat{y}_{Ai}$  (或  $\hat{y}_{Bi}$ ) 是输入为  $\mathbf{X}_{Ai}$  (或  $\mathbf{X}_{Bi}$ ) 时的理想输出  $M_A$  和  $M_B$  分别为两个训练集样本的个数。设用训练集  $A$  和  $B$  分别训练两个结构完全相同的网络，训练完成后二者的映射函数分别是  $y = f_A(\mathbf{X})$  以及  $y = f_B(\mathbf{X})$ 。则无偏准则  $UC$  定义如下：

$$UC = \left\{ \sum_{i=1}^{M_A} [f_A(\mathbf{X}_{Ai}) - f_B(\mathbf{X}_{Ai})]^2 + \sum_{i=1}^{M_B} [f_A(\mathbf{X}_{Bi}) - f_B(\mathbf{X}_{Bi})]^2 \right\}^{\frac{1}{2}} \quad (5-112)$$

与此相似的是规则化准则  $RC$  (regularity criterion 的缩写)，其定义如下：

$$RC = \left\{ \frac{1}{M_A} \sum_{i=1}^{M_A} [\hat{y}_{Ai} - f_B(\mathbf{X}_{Ai})]^2 + \frac{1}{M_B} \sum_{i=1}^{M_B} [f_A(\mathbf{X}_{Bi}) - \hat{y}_{Bi}]^2 \right\} \quad (5-113)$$

另外一种计算较简单的准则是  $SUC$  (simplified unbiasedness criterion) 其定义如下：

$$SUC = \frac{1}{M_B} \sum_{i=1}^{M_B} \frac{|f_A(\mathbf{X}_{Bi}) - \hat{y}_{Bi}|}{|\hat{y}_{Bi}|} \quad (5-114)$$

下面以  $RC$  为例，介绍其在系统辨识和控制中的应用。

## 5.6.2 FNN 在系统辨识中的应用举例

这一节以煤气炉的动态过程<sup>[37]</sup>为例来讨论有关的各项问题 (模拟研究)

### 1. 问题的提出

设有输入序列  $u(\nu)$  和输出序列  $y(\nu)$  分别为煤气炉的气流速度和  $\text{CO}_2$  集中度。若已知  $u(\nu), u(\nu-1), \dots$  和  $y(\nu), y(\nu-1), \dots$  要求网络给出  $y(\nu+1)$  的尽可能准确预测值。文献 [37] 给出了一个样本集  $\{u(\nu), y(\nu)\}, \nu = 1 \sim 296$  其中  $\nu = 1 \sim 254$  用于网络训练，后 42 个样本用来对训练后的网络进行测试。测试按一步预测 (见 2.13.1 小节的 (3) 及 2.14.3 小节的 (1)) 和多步预测 (见 2.13.1 小节的 (4) 及 2.14.3 小节的 (2)) 两种方式进行。前者按  $y(\nu), y(\nu-1), \dots$  可获取的情况运行；后者按  $y(\nu), y(\nu-1), \dots$  不可获取，只能用网络产生的输出估计值  $\hat{y}(\nu), \hat{y}(\nu-1), \dots$  来替代的情况运行。由于  $\hat{y}(\nu)$  与  $y(\nu)$  不可能完全一致，后者的测试误差大于前者。

### 2. 方案

为了用一尽可能简单的 FNN 规则数尽可能少实现上述的系统辨识任务 采取下列方案。第一 用 GMDH 技术和 ST (search tree) 法从各输入变量中选择最关键的变量。文献 [12] 以  $y(\nu), u(\nu-2), u(\nu-3), u(\nu-4)$  作为输入变量的候选者来进行选择 (其他研究者的选择将在下面 5 项中介绍) 本节只讨论这一选择过程。第二 网络采取一阶 TSK 模型 (见 5.2.2 小节)。输入和输出空间的划分以及规则的建立采用 MMC 算法 (见 5.3.2 小节的 5) 其中最关键的参数是一般称为影响半径的  $r_a$  (见式 (5-30)),  $r_a$  越小则规则数多

且逼近误差小，反之则规则数少但逼近误差大。TSK 模型中隶属函数的参数用 BP 算法细调（见 5.3.4 小节）线性组合系数用 RLSE 算法决定（见 5.3.6 小节）在模拟研究中按照  $r_a$  的不同取值，用 GMDH-ST 算法求出各  $r_a$  相应的最优选择输入变量（最优是指相应的 RC 值最小）并且用 MMC, BP, RLSE 算法求出相应的网络结构（输入和输出空间划分、规则数及推理规则）及各项参数。

### 3. GMDH-ST 算法介绍

设 4 个待选者表示为  $U_1 = y(\nu), U_2 = u(\nu - 2), U_3 = u(\nu - 3), U_4 = u(\nu - 4)$ 。ST 如图 5-10 所示 其中 0 层只包含一个节点  $\phi$  它是搜索的出发点 设有  $Q$  个待选者 此例中  $Q = 4$  则搜索从 1 层至  $Q$  层逐层向下进行。第 1 层的节点数为  $Q! / ((Q - 1)! \times 1!)$  第 2 层的节点数为  $Q! / ((Q - 2)! \times 2!)$  等等。第 1 层的每个节点表示只选该变量作为网络输入并且在  $r_a$  固定的条件下求得网络的结构和参数。这样，每个节点可产生一个网络并且用 GMDH 法求得其 RC 值。然后，从 RC 值最小的节点出发，对第 2 层进行搜索。例如，第 1 层中 RC 值最小节点为  $U_1$  则第 2 层中只对  $(U_1, U_2), (U_1, U_3), (U_1, U_4)$  这 3 个节点进行搜索。如此可逐层下推 直至找到一个节点的 RC 是所有节点中最小者。该节点相应的各变量即是给定  $r_a$  条件下的最优选变量，相应的 RC 值记为  $RC_{\min}$ 。对于不同的  $r_a$  值 重复上述搜索过程，则可以求得各自相应的 RC 值 最优选变量和规则数。设计者即可根据需要 择定其中的某一种。按 ST 算法进行搜索的优点是当  $Q$  值较大时，使搜索量大幅下降<sup>[12]</sup>

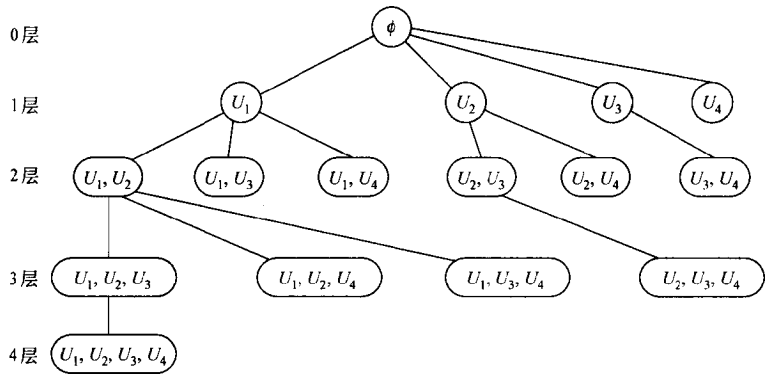


图 5-10 ST 的搜索示例

### 4. 模拟实验结果介绍

搜索结果表明 当  $r_a = 0.35$  时，图 5-10 第 3 层的  $(U_1, U_3, U_4)$  节点所相应的 RC 值为 0.153 较其他任何  $r_a$  所得到的 RC 值更小，因此网络选  $y(\nu), u(\nu - 3), u(\nu - 4)$  为前提变量。在此条件下，再对不同的网络结构和性能进行考查，表 5-1 列出 3 组不同结构的性能。

可以看到，1, 2 两组结构中，经过参数学习后第 2 组的性能优于第 1 组。作为对比，第 3 组用了两个输入变量和 8 条规则，其复杂度远高于第 2 组且用训练集内数据测得的均方误差小于第 2 组，但是用训练集外数据测得的均方误差大于第 2 组。这说明第 3 组的推广性能不如第 2 组。

表 5-1 3 组不同结构的性能比较

序号	$r_a$	RC /min	TSK 模型 输入变量	$MSE^*$ (多步)	$MSE^*$ (单步)	规则 数	参数 个数	迭代 次数	步幅	$MSE^*$ (多步)	$MSE^*$ (单步)	$MSE^{**}$ (多步)	$MSE^{**}$ (单步)
1	0.85	0.162	$y(\nu), u(\nu-3), u(\nu-4)$	0.86	0.15	3	30	500	0.50	0.892	0.128	0.675	0.194
2	0.50	0.162	$y(\nu), u(\nu-3), u(\nu-4)$	0.867	0.152	3	30	500	0.50	0.811	0.112	0.267	0.065
3	0.30	0.165	$y(\nu), u(\nu-3)$	0.779	0.101	8	80	500	0.50	0.754	0.093	0.302	0.083
结构学习结果								参数学习结果					

说明  $MSE^*$  : 用训练集内数据按多步预测方式工作的均方误差 ;  
(多步)

$MSE^*$  : 用训练集内数据按单步预测方式工作的均方误差 ;  
(单步)

$MSE^{**}$  , 用训练集外数据测试所得的均方误差。

第 2 组网络的 3 条规则可以写成

$R^1$ : IF  $y(\nu)$  is  $A_1^1$  and  $u(\nu-3)$  is  $A_2^1$  and  $u(\nu-4)$  is  $A_3^1$   
THEN  $y(\nu+1) = 2.532y(\nu) + 0.678u(\nu-3) + 2.702u(\nu-4) - 82.1$

$R^2$ : IF  $y(\nu)$  is  $A_1^2$  and  $u(\nu-3)$  is  $A_2^2$  and  $u(\nu-4)$  is  $A_3^2$   
THEN  $y(\nu+1) = 1.984y(\nu) - 1.57u(\nu-3) + 3.879u(\nu-4) - 45.09$

$R^3$ : IF  $y(\nu)$  is  $A_1^3$  and  $u(\nu-3)$  is  $A_2^3$  and  $u(\nu-4)$  is  $A_3^3$   
THEN  $y(\nu+1) = 0.755y(\nu) - 1.504u(\nu-3) + 0.494u(\nu-4) + 12.76$

$A_1^i, A_2^i, A_3^i$  所相应的隶属函数  $\mu_{A_1^i}(y(\nu)), \mu_{A_2^i}(u(\nu-3)), \mu_{A_3^i}(u(\nu-4)), i = 1 \sim 3$  分别如图 5-11(a) ~ (c) 所示。各图中标数为 1 ~ 3 的曲线相应于  $i = 1 \sim 3$  的隶属函数。

其中每个隶属函数为高斯形，用均值  $M$  和方差  $\sigma$  两个参数定义 每条规则含 3 个前提隶属函数故有 6 个参数。再加上结论线性组合中的 4 个系数 每条规则含 10 个参数。第 2 组网络给出的单步及多步预测结果如图 5-12 所示，其中前 254 点是训练集，后 42 点是测试集，(a) 为单步预测结果，(b) 为多步预测结果。图中  $y(\nu+1)$  是实际系统输出值， $\hat{y}(\nu+1)$  是预测的系统输出值。

## 5. 其他模拟实验结果

由于煤气炉问题是一个经典的系统辨识问题<sup>[39]</sup> 很多算法都以为之准绳来比较各自的模拟实验结果。下面给出几个都是基于同一问题且都采用 1 阶 TSK 模型时的模拟实验结果。文献[20]用  $y(\nu)$  和  $u(\nu-4)$  作为输入变量 规则数为 5，单步预测时训练集内的均方误差值为 0.158。文献[40]用  $y(\nu), y(\nu-1), y(\nu-2), u(\nu-1), u(\nu-2), u(\nu-3)$  作为输入变量，规则数为 2，单步预测时训练集内的均方预测误差为 0.068。文献[41]用  $y(\nu), u(\nu-3), u(\nu-4)$  作为输入变量，规则数为 6，单步预测时训练集内均方预测误差为 0.190。

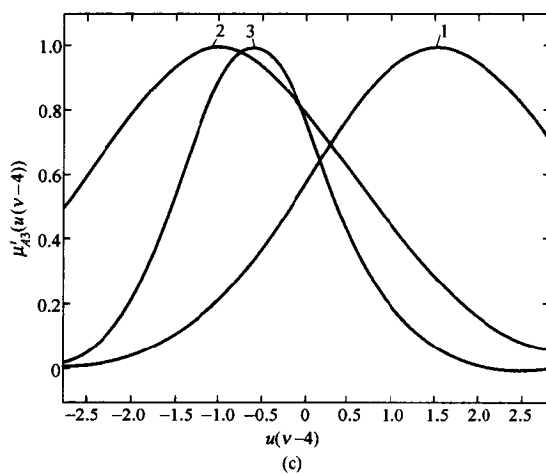
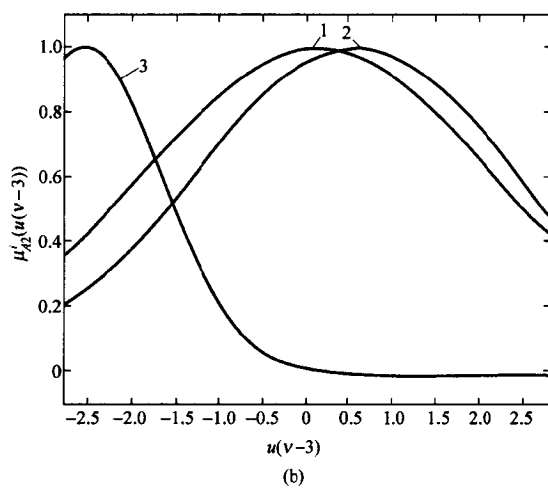
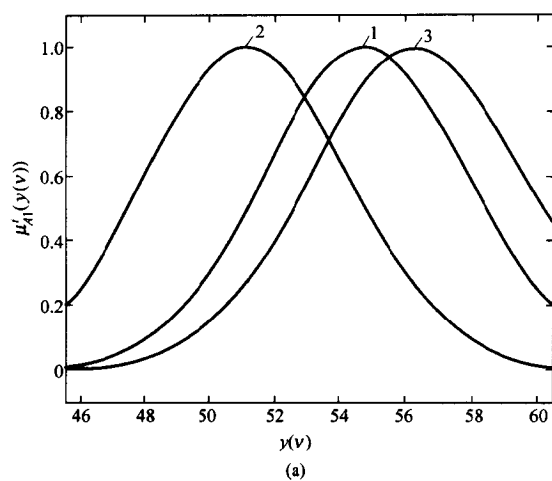


图 5-11 隶属函数  $\mu_A^i$  曲线



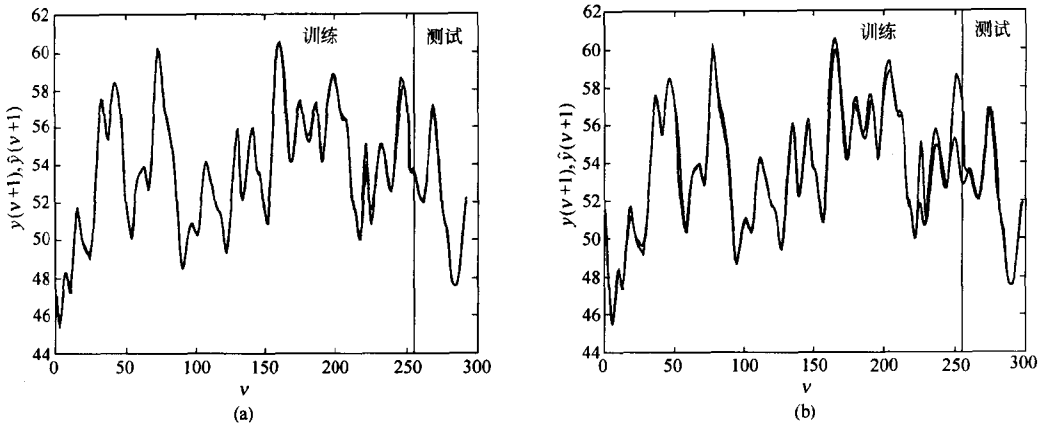


图 5-12 第 2 组网络的单步和多步预测结果

### 5.6.3 FNN 在系统控制中的应用举例

本小节讨论一个较简单的控制问题。设有一个 SISO 非线性系统 其输出  $y(v)$  和输入  $u(v)$  之间的关系用下列方程描述：

$$y(v+1) = \hat{g}[y(v), y(v-1), \dots, y(v-P)] + u(v)$$

其中  $\hat{g}[\cdot]$  是一个未知的非线性函数,  $P$  也是未知的 (这个问题之所以简单是因为  $y(v+1)$  和  $u(v)$  之间具有线性关系)。待完成的任务是, 如要求输出为  $y^*(v+1)$  求  $u(v)$ 。如果  $\hat{g}[\cdot]$  和  $P$  为已知且  $y(v), y(v-1), \dots, y(v-P)$  是可测量的, 则易于求得

$$u(v) = y^*(v+1) - \hat{g}[y(v), y(v-1), \dots, y(v-P)]$$

如  $\hat{g}[\cdot]$  和  $P$  为未知, 则应利用一个训练集对其进行估计。文献 [42]、[43] 给出了下面的一个实例：

$$\hat{g}[y(v), y(v-1)] = \frac{y(v)y(v-1)[y(v)+2.5]}{1+y^2(v)+y^2(v-1)}$$

文献 [12] 给出了一个训练集  $\{u(v), y(v)\}$ ,  $v = 1 \sim 400$  其中当  $v = 1 \sim 200$  时  $u(v)$  是在  $[-2, 2]$  间隔内均匀分布的随机变量；当  $v = 201 \sim 300$  时  $u(v) = \sin(2\pi v/25)$ ；当  $v = 301 \sim 400$  时  $u(v) = 0.5\sin(2\pi v/25) + 0.5\sin(2\pi v/250)$ 。前 200 点数据分成 A、B 两组 每组 100 点 按上列 5.6.2 小节所述方法来选择最优输入变量、确定网络结构及调整参数。网络待逼近的输入-输出关系可以写成

$$z(v) = y(v+1) - u(v) = \hat{g}[y(v), y(v-1)]$$

训练集可以写成  $\{z(v), y(v)\}$ , 网络映射函数可表示为  $z(v) = \hat{g}[y(v), y(v-1), \dots, y(v-P)]$ 。采用 1 阶 TSK 模型及 5.6.2 小节的方法, 可以很容易确定优选输入变量为  $y(v)$  和  $y(v-1)$  (候选者为  $y(v), y(v-1), y(v-2), y(v-3)$ ) 最优结构有两组, 一组取  $r_a = 0.4, RC_{\min} = 0.033$  规则数等于 4 参数个数等于 28, 多步集内均方预测误差值  $MSE^* = 0.313$  另一组取  $r_a = 0.30, RC_{\min} = 0.014$  规则数等于 8 参数个数等于 56, 多步集内均方预测误差值为  $MSE^* = 0.015$ 。训练集中后面各 100 点数据分别用来检

验已建模型的预测精确度。在 多步预测的条件下，第一个 100 点对上述两组结构的 MSE 值分别为 0.0031 和 0.005 第二个 100 点的 MSE 值分别为 0.00081 和 0.0063。整个控制系统的构成如图 5-13 所示。有关 FNN 在控制领域中的应用问题还可参阅文献<sup>[26,44,38]</sup>。

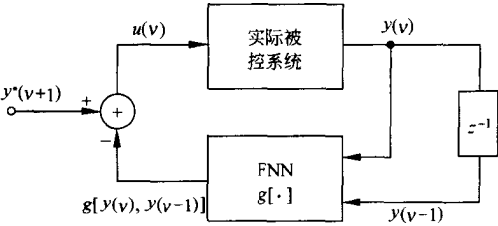


图 5-13 控制系统的构成

### 5.6.4 增强学习问题

在一般的控制问题中，应用 FNN 时可以取得若干对训练样本  $\{X_p, \hat{y}_p\}, p = 1 \sim P$  构成的训练集，它们描述了网络的输入至理想输出之间的映射关系。这就是网络进行监督学习的依据，通过学习可以使网络的实际输出  $y_p$  与  $\hat{y}_p$  之间的均方差值达到最小（为简化起见，这里以 MISO 系统为例来讨论，下同）但是在实际

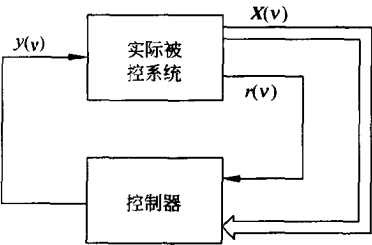


图 5-14 控制系统原理图

控制问题中，常遇到另一种情况，这时难以获得明确的训练集，而按照图 5-14 所示的原理来进行工作。设被控系统和控制器都按照离散时间  $v$  运行。被控系统接收控制信号  $y(v)$ ，提供关于系统状况的状态向量  $X(v)$  以及一个描述运行结果是否良好的信号  $r(v)$ 。在最简单的情况下， $r(v)$  只能取两种值：若  $r(v) = 0$ （或 1）表示结果良好，称为奖信号；若  $r(v) = -1$  表示不好，称为惩信号。较复杂的情况是  $r(v)$  能在间隔  $[0, -1]$  内取若干离散值或连续取值， $r(v)$  越接近 0 时，结果越好；越接近  $-1$  时，结果越坏。控制器的设计 requirements 是根据系统提供的  $X(v)$  产生控制信号  $y(v)$ ，使得  $r(v)$  的积累值达到极大，即达到最佳运行结果。 $r(v)$  即称为增强信号（reinforcement signal）。考虑到系统中存在着由输入到结果之间的延迟，分单步及多步延迟两种情况来讨论。在单步延迟情况下，被控系统在时刻  $v$  的状态  $X(v)$  取决于  $(v-1)$  时刻的状态  $X(v-1)$  以及系统输入  $y(v-1)$ 。 $v$  时刻的增强信号  $r(v)$  也取决于  $X(v-1)$  和  $y(v-1)$ 。控制器给出的  $y(v-1)$  不是为了达到某个特定的  $X(v)$ ，而是使  $r(v)$  尽量大，而且控制器必须在实际运行环境中学得如何由被控系统提供的  $X(v-1)$  来产生所需的  $y(v-1)$ ，这就是增强学习问题。这里的难处在于采集不到若干对训练样本  $\{X_p, \hat{y}_p\}$  供控制器学习时使用。在多步延迟的情况下困难更大，这时  $r(v)$  取决于若干时刻前的输入控制信号  $y(v-\rho), y(v-\rho+1), \dots, y(v)$ ， $\rho$  是一个大于 1 的未知整数。为了解决增强学习问题，必须采取具有预测机制的学习结构，其简略框图如图 5-15 所示。其中 FP 表示一个由 FNN 构成的预

测器,它根据  $X(\nu-1)$  给出  $r(\nu)$  的估值  $p(\nu-1)$ ; FLC 表示一个由 FNN 构成的控制器,它根据  $X(\nu-1)$  给出  $y(\nu-1)$ 。图中的虚线表示 FLC 在进行学习时的信号流向。FP 通过学习,使预测增强信号  $p(\nu-1)$  和实际增强信号  $r(\nu)$  之间的均方误差达到最小。FLC 利用  $p(\nu-1)$  和  $r(\nu)$  进行学习,使得由  $y(\nu-1)$  输送到被控系统后,该系统所产生的实际增强信号  $r(\nu)$  达到最大。下面将分别就单步延迟和多步延迟的情况讨论 FP 和 FLC 的学习问题。

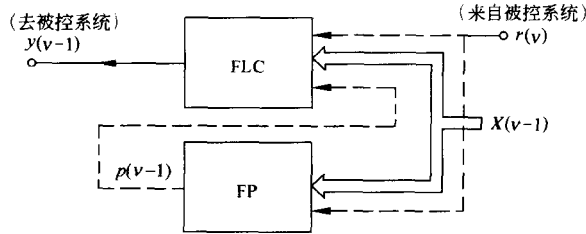


图 5-15 具有预测机制的学习结构框图

具有预测能力的增强学习在系统动态控制,人工智能等许多领域中的应用较之简单的监督学习要宽广得多。从近代生物学习理论(生物学、认知学等)的角度看,增强学习也较监督学习有用得多<sup>[46]</sup>。举人学习驾车的过程为例可以说明这一点,若不出事故则  $r(\nu) = 0$ ,若出事故则  $r(\nu) = -1$ 。人根据环境作出驾车动作时应使  $r(\nu)$  的积累值达到最大。人在学习驾车取得经验时,对  $r(\nu)$  值的预期将起很大作用。再举一个下棋的例子,被控系统的状态即是棋盘上的棋局,控制者走的每一步棋即为控制信号。但是,增强信号不可能每走一步棋就可从被控系统返回,而是要经过很多步直至终盘时才见分晓。而此例的复杂性更在于棋局不但由我方而且也由对方操纵的,因此其不确定性非常大。为了学习如何赢棋,一个预测机构更是必要,它能够通过熟悉对手的棋风和思路,根据棋局形势作出终局胜负的判断。

### 1. 具有单步预测的增强学习

整个控制器由 FP 和 FLC 两个 FNN 组成(见图 5-15)。其中 FP 的输入是  $X(\nu-1)$ ,其输出  $p(\nu-1)$  为仅根据  $X(\nu-1)$  (不考虑被控系统接收的控制信号)而作出的对  $r(\nu)$  的预测值。FP 的学习目的是通过调整网络中的各个权值(假设网络结构已经确定),使得  $E(\nu) = [r(\nu) - p(\nu-1)]^2$  达到最小。在单步预测时,每一时刻都能收到回馈的增强信号  $r(\nu)$ ,因此按照最陡下降原理,可求得网络中任意一个权  $w$  的下列调整公式:

$$w(\nu) = w(\nu-1) + \Delta w(\nu-1)$$

$$\Delta w(\nu-1) = -\alpha(\nu) \left. \frac{\partial E(\nu)}{\partial w} \right|_{w=w(\nu-1)} = 2\alpha(\nu)[r(\nu) - p(\nu-1)] \left. \frac{\partial p(\nu-1)}{\partial w} \right|_{w=w(\nu-1)} \quad (5-115)$$

其中  $\partial p(\nu-1)/\partial w$  可以用常规的 BP 算法计算。 $\alpha(\nu)$  是一个随  $\nu$  增而逐渐下降的步幅函数。

为了讨论 FLC 的学习,将图 5-15 的两个 FNN 更细致地描绘图 5-16 中。在此图所示结构中,两个 FNN 共用 1~3 层,其作用是计算输入向量  $X(\nu-1)$  相对于  $q$  个输入模糊空间的  $q$  个隶属函数值  $\mu_A^i(X(\nu-1))$ ,  $i = 1 \sim q$ 。FP-FNN 和 FLC-FNN 使用各自的 4~5 层,

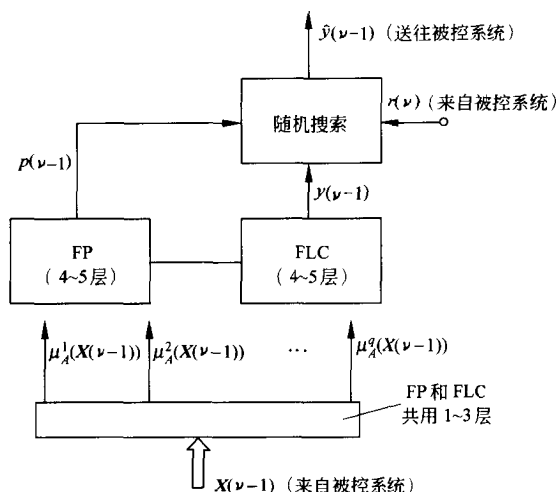


图 5-16 图 5-15 中 FNN 的详细示意图

前者的输出为  $p(v-1)$  后者的输出  $y(v-1)$  为送往被控系统的控制信号。FLC 的学习目的是调整其中的各个权  $w$  使得  $r(v)$  达到极大。由于此学习不具备监督学习所需的理想输出，所以采取下列的随机搜索方案（在图 5-16 中由“随机搜索”方块来完成）

设  $v-1$  时刻 FLC 的输入和输出分别是  $X(v-1)$  和  $y(v-1)$ 。在进行随机搜索方式的学习时，不将  $y(v-1)$  直接送到被控系统作为控制信号，而叠加一个均值为 0、方差为  $\sigma^2(v-1)$  的随机变量  $\Delta y(v-1)$  形成  $\hat{y}(v-1) = y(v-1) + \Delta y(v-1)$  来作为控制信号。其中  $\sigma(v-1)$  按下式计算：

$$\sigma(v-1) = \frac{K}{1 + \exp[2p(v-1)]} \quad (5-116)$$

其中  $K$  是一个常数（例如，取  $K = 1$ ）。若  $p(v-1) = -1$ （惩），则  $\sigma(v-1) \approx K$ ；若  $p(v-1) = 1$ （奖），则  $\sigma(v-1) \approx 0.12K$ 。前者使  $\hat{y}(v-1)$  在偏离  $y(v-1)$  较大的范围内搜索；后者则搜索范围很小。为了按最陡下降算法调整 FLC 中的各个权  $w$  使得  $r(v)$  极大，应按下式进行计算：

$$w(v) = w(v-1) + \Delta w(v-1)$$

$$\Delta w(v-1) = \alpha(v) \left. \frac{\partial r(v)}{\partial w} \right|_{w=w(v-1)} = \alpha(v) \frac{\partial r(v)}{\partial y(v-1)} \cdot \left. \frac{\partial y(v-1)}{\partial w} \right|_{w=w(v-1)} \quad (5-117)$$

式(5-117)最右侧的偏微分可用常规 BP 算法计算，而其前一项可用下式近似计算：

$$\frac{\partial r(v)}{\partial y(v-1)} \approx [r(v) - p(v-1)] \left[ \frac{\hat{y}(v-1) - y(v-1)}{\sigma(v-1)} \right] \quad (5-118)$$

## 2. 具有多步预测的增强学习

在实际问题中，一个被控系统在时刻  $v$  的状态  $X(v)$  和增强信号  $r(v)$  不仅依赖于前一时刻的系统输入  $y(v-1)$  还依赖于  $y(v-2), \dots, y(v-\rho)$ ， $\rho$  是一个未知正整数且此依赖关系有相当大的不确定性。更有甚者，被控系统往往不是每个时刻  $v$  都给出增强信号，而只有在 FLC 将一长串控制信号加于被控系统以后，才能获得它提供的一个增强信号（例

如弈棋)这使得在 FLC 的学习过程中,目标函数的建立更困难。因为 FLC 每一时刻给出的控制命令必须基于对最终的奖/惩结果而作出,在最终结果没有出来以前则只能根据预测来作决定。这就需要一个多步模糊预测器(FP),它的任务是在第一个增强信号来到之后而第二个增强信号尚未来到之前的中间若干个时刻,对将来到的增强信号作出预测。由于每个时刻给出的都是预测的增强信号,FP 和 FLC 要根据此预测作出自身参数的调整,而不仅在最后实际增强信号来到时才调整。下面介绍的时差法(temporal difference method)与动态规划法相关联<sup>[47~50]</sup>。在单步预测 FP 的学习中,目标函数为实际增强信号与预测增强信号差值的平方(与监督学习相同)。在多步预测 FP 的学习中,目标函数为本时刻预测增强信号与上一时刻预测增强信号差之平方,此即时差法。下面分三种情况介绍 FP 的时差学习算法。然后介绍 FLC 的相应学习算法。

#### (1) FP 时差学习 —— 情况 1

设于时刻  $\nu = 0$ , 被控系统给出了实际增强信号  $r(0)$ , 而于  $\nu = 1 \sim m$  未给出增强信号, 又于  $\nu = m+1$  时再次给出增强信号  $r(m+1)$ 。现讨论 FP 于  $\nu = 1 \sim m+1$  诸时刻如何进行时差学习。设系统在  $\nu = 0 \sim m$  诸时刻的状态为  $\mathbf{X}(0) \sim \mathbf{X}(m)$ , 而 FP 在这些时刻提供的关于将要出现的实际增强信号预测值为  $p(0) \sim p(m)$ 。又设 FP 中所有权值构成向量  $\mathbf{W}$ , 那么在上列时刻的权可表示为  $\mathbf{W}(0) \sim \mathbf{W}(m)$ 。  $p(\nu) = p[\mathbf{W}(\nu), \mathbf{X}(\nu)]$ ,  $\nu = 0 \sim m$ 。设  $m+1$  时刻出现的实际增强信号为  $r(m+1)$ 。那么可以按下列规则来调整 FP 的权向量  $\mathbf{W}$ :

$$\begin{aligned} \mathbf{W}(\nu) &= \mathbf{W}(\nu-1) + \Delta\mathbf{W}(\nu-1), \quad \nu = 1, 2, \dots, m+1 \\ \Delta\mathbf{W}(\nu-1) &= \begin{cases} \alpha[p(\nu) - p(\nu-1)] \sum_{k=1}^{\nu-1} \lambda^{\nu-k-1} \nabla_{\mathbf{W}} p(k), & \nu = 1, \dots, m \\ \alpha[r(\nu) - p(\nu-1)] \sum_{k=1}^{\nu-1} \lambda^{\nu-k-1} \nabla_{\mathbf{W}} p(k), & \nu = m+1 \end{cases} \end{aligned} \quad (5-119)$$

其中  $0 \leq \lambda \leq 1$ ,  $0 < \alpha < 1$  为步幅。 $\nabla_{\mathbf{W}} p(k)$  可用常规 BP 算法计算。若  $\lambda = 1$  则  $\nu$  时刻的权调整量取决于  $k = 1 \sim \nu-1$  诸时刻的梯度之和与预测差值的乘积; 若  $\lambda < 1$  则  $k$  越小时相应的梯度对  $\mathbf{W}(\nu)$  影响越小; 若  $\lambda = 0$  则只有  $\nu-1$  时刻的梯度影响  $\mathbf{W}(\nu)$ 。这种方案称为最终(增强)输出预测 final prediction of outcome 算法。

#### (2) FP 时差学习 —— 情况 2

设在时刻  $\nu = 1, 2, \dots, m+1$ , 实际系统给出的增强信号为  $r(\nu)$ 。在这些时刻上  $r(\nu)$  可能到达(这时  $r(\nu)$  有一定取值), 也可能没有到达(这时  $r(\nu) = 0$ )。设  $m$  是一个固定的正整数(有限值), 设  $z(\nu-1)$  为增强信号  $r(k)$  在  $k = \nu \sim m+1$  间隔内的积累值, 即

$$z(\nu-1) = \sum_{k=\nu}^{m+1} r(k), \quad \nu = 1 \sim m+1 \quad (5-120)$$

令 FP 的输出  $p(\nu-1)$  为  $z(\nu-1)$  的预测值, 且令

$$p(m+1) = 0 \quad (5-121)$$

这样, 预差误差  $z(\nu-1) - p(\nu-1)$  可表示为

$$z(\nu-1) - p(\nu-1) = \sum_{k=\nu}^{m+1} [r(k) + p(k) - p(k-1)], \quad \nu = 1 \sim m+1 \quad (5-122)$$

FP 的权向量调整公式如下:

$$W(\nu) = W(\nu-1) + \Delta W(\nu-1)$$

$$\Delta W(\nu-1) = \alpha[r(\nu) + p(\nu) - p(\nu-1)] \sum_{k=1}^{\nu-1} \lambda^{\nu-k-1} \nabla_w p(k), \quad \nu = 1 \sim m+1 \quad (5-123)$$

其中  $0 \leq \lambda \leq 1, 0 < \alpha < 1$ 。

此情况发生于对未来一段时间内增强信号之和感兴趣而并非只对某次增强信号来到感兴趣的场合。所以称之为有限输出积累预测(Prediction of finite cumulative outcomes)。

### (3) FP 时差学习 —— 情况 3

设  $z(\nu-1)$  为从时刻  $\nu$  开始至  $\infty$  的所有增强信号的“折扣和”其定义如下：

$$z(\nu-1) = \sum_{k=0}^{\infty} \gamma^k r(\nu+k) \quad (5-124)$$

其中  $0 \leq \gamma < 1$  称为折扣(discount)系数。现在用 FP 的输出  $p(\nu-1)$  来预测  $z(\nu-1)$ 。注意当折扣系数  $\gamma$  接近于 1 时,  $p(\nu-1)$  预测的是从  $\nu$  开始的很长一串增强信号之和; 当  $\gamma$  接近于 0 时只是接近和等于  $\nu$  的增强信号之和。可以看到,  $z(\nu-1) = r(\nu) + \gamma z(\nu) \approx r(\nu) + \gamma p(\nu)$ 。这样,  $z(\nu-1)$  和  $p(\nu-1)$  之间误差的近似计算式为

$$z(\nu-1) - p(\nu-1) \approx r(\nu) + \gamma p(\nu) - p(\nu-1)$$

为了使此预测误差达到极小, 每个时刻  $\nu$  的权应按下式调整:

$$W(\nu) = W(\nu-1) + \Delta W(\nu-1)$$

$$\Delta W(\nu-1) \approx \alpha[r(\nu) + \gamma p(\nu) - p(\nu-1)] \sum_{k=1}^{\nu-1} \lambda^{\nu-k-1} \nabla_w p(k), \quad \nu \geq 2 \quad (5-125)$$

其中  $0 < \alpha < 1$  为步幅。 $0 \leq \lambda \leq 1$ 。当  $\lambda$  趋近于 1 时 权调整量与时刻 1 至  $\nu-1$  诸  $p(k)$  梯度之和成正比 当  $\lambda$  趋近 0 时, 调整量只与  $\nu-1$  时刻之梯度成正比。这种情况称为无限折扣输出积累预测(prediction of infinite discounted cumulative outcomes), 它适用于只有经过很多步预测后才会有实际增强信号到来的情况。

### 3. FLC 的时差学习

与单步学习情况相似, 以上述情况 3 为例, FLC 中每一个权的调整量可用式 (5-117) 计算, 只是其最右侧的偏微分由原式 (5-118) 改为

$$\frac{\partial r(\nu)}{\partial y(\nu-1)} \approx \left[ \frac{\partial r(\nu)}{\partial y(\nu-1)} + \gamma p(\nu) - p(\nu-1) \right] \left[ \frac{y(\nu-1) - p(\nu-1)}{\sigma(\nu-1)} \right] \quad (5-126)$$

### 4. 用 1 阶 TSK 模型实现增强学习时, 其中的结构/参数学习的实施方案

在 (1)~(3) 中讨论了 FP 或 FLC 在无法取得明显监督信号时如何求得权调整量的计算公式, 但全部结构/参数学习过程仍不明确。下面介绍基于 1 阶 TSK 模型的实施方案<sup>[45]</sup> 而基于 Fuzzy ART 的实施方案可参考文献[47], 此处不叙述。这里介绍的学习过程流程图如图 5-17 所示。下面依次介绍其中各模块的功能。

#### (1) 初始化

在学习开始前, 须首先给出输入和输出空间的模糊划分数  $q$  和  $R$  以及相应的隶属函数  $\mu_A^i(\mathbf{X}), i=1 \sim q$  和  $\mu_B^l(y), l=1 \sim R$  (见 5.2.1)。此外, 还要给出  $R$  条推理规则 即给定各  $\mu_A^i(\mathbf{X})$  与  $\mu_B^l(y)$  之间的连接关系 (见式 (5-2)~式 (5-4))。这一切在初始化模块中由人工完成, 主要的初始化如下。第一, 将  $\mathbf{X}$  的每一维  $x_o$  用  $k_n$  个均匀间隔的模糊数表示,

其隶属函数为  $\mu_{A_n}^{i_n}(x_n)$ ,  $i_n=1\sim k_n$ ,  $k_n$  的大小由人工决定。这些隶属函数的中心值在  $x_n$  取值区间中均匀排列, 相邻函数有一定覆盖, 函数取高斯形(式 5-20))。  $\mu_A^i(\mathbf{X})$  由  $N$  个  $\mu_{A_n}^{i_n}(x_n)$  相乘而得(式 5-16))。  $q=k_1\times k_2\times\cdots\times k_N$ 。各输入隶属函数的参数在学习过程中调整, 但是各  $k_n$  不再改变。在学习结束时,  $q$  个  $\mu_A^i(\mathbf{X})$  中有若干个可能始终未被启用过则可予以废除。第二,  $R$  的大小也由人工决定, 各  $\mu_B^l(y)$  在  $y$  取值区间均匀排列。  $R$  值可以在结构学习中改变(增加), 各  $\mu_B^l(y)$  参数在参数学习中调整。第三, 给出各  $\mu_B^l(y)$  与各  $\mu_A^i(\mathbf{X})$  在推理规则中的连接关系。在没有任何先验知识时可取随机连接方式。最好能取得已有的专家经验来建立初始连接, 这会加快学习进程。在结构学习中连接是可以改变的。

(2) 前向计算、时差预测和随机搜索计算

见上列 1、2、3 诸小节。

(3) 进行结构学习否? 加新节点否?

设于时刻  $\nu-1$  在网络的第 4 层已建立了  $R(\nu-1)$  个模糊区, 每个区与一条推理规则的结论对应, 其隶属函数为  $\mu_B^l(y) = \exp\left[-\left(\frac{y-b_0^l(\nu-1)}{\sigma_0^l(\nu-1)}\right)^2\right]$ ,  $l=1\sim R(\nu-1)$ 。按增强学习和 BP 算法, 基于式(5-48)、(5-49)和(5-118)在单步预测情况下 FLC 第 4 层的各隶属函数参数  $b_0^l, \sigma_0^l$  可按照下式进行调整:

$$\begin{cases} b_0^l(\nu) = b_0^l(\nu-1) + \Delta b_0^l(\nu-1) \\ \sigma_0^l(\nu) = \sigma_0^l(\nu-1) + \Delta \sigma_0^l(\nu-1), l=1\sim R(\nu-1) \end{cases}$$

$$\Delta b_0^l(\nu-1) = \alpha[r(\nu) - p(\nu-1)] \left\{ \frac{\hat{y}(\nu-1) - y(\nu-1)}{\sigma(\nu-1)} \right\} \frac{\sigma_0^l(\nu-1)\mu^l(\nu-1)}{\sum_{u=1}^{R(\nu-1)} \sigma_0^u(\nu-1)\mu^u(\nu-1)} \quad (5-127)$$

$$\Delta \sigma_0^l(\nu-1) = \alpha[r(\nu) - p(\nu-1)] \left\{ \frac{\hat{y}(\nu-1) - y(\nu-1)}{\sigma(\nu-1)} \right\} \times \frac{\mu^l(\nu-1) \left[ b_0^l(\nu-1) \sum_{u=1}^{R(\nu-1)} \sigma_0^u(\nu-1)\mu^u(\nu-1) - \sum_{u=1}^{R(\nu-1)} b_0^u(\nu-1)\sigma_0^u(\nu-1)\mu^u(\nu-1) \right]}{\left[ \sum_{u=1}^{R(\nu-1)} \sigma_0^u(\nu-1)\mu^u(\nu-1) \right]^2} \quad (5-128)$$

其中  $\alpha$  为步幅,  $\mu^l(\nu-1)$  是输入为  $\mathbf{X}(\nu-1)$  时第  $l$  条规则成立的隶属函数(参见式(5-3))。对于多步预测和 FP 网络也可依同理计算。

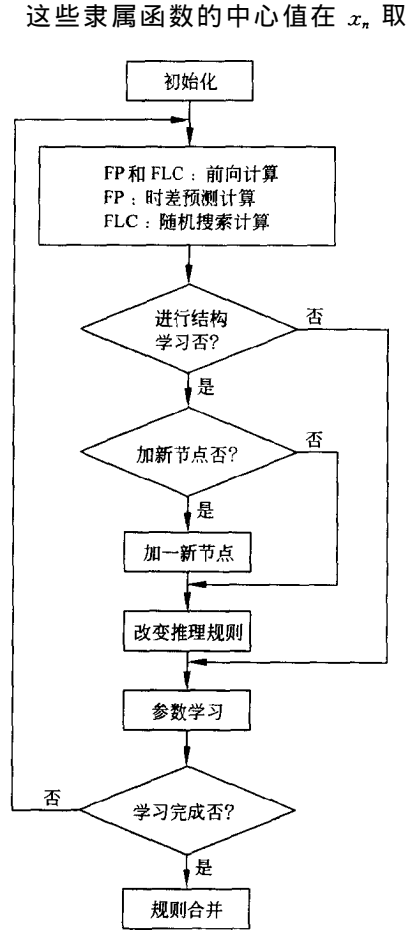


图 5-17 学习过程流程图

设进行参数调整后的隶属函数是  $\mu_{B'}^l(y) = \exp\left[-\left(\frac{y-b_0^l(\nu)}{\sigma_0^l(\nu)}\right)^2\right]$ ,  $l'=1 \sim R(\nu-1)$  而调整前的隶属函数  $\mu_{B'}^l(y)$  已在前面列出。现在计算每一个  $\mu_{B'}^l(y)$  与  $\mu_{B'}^l(y)$ ,  $l=1 \sim R(\nu-1)$  的相似度  $E(\mu_{B'}^l, \mu_{B'}^l)$ , 其定义及计算公式见式 5-25) (5-28)。设

$$D(l') = \max_{l=1 \sim R(\nu-1)} [E(\mu_{B'}^{l'}, \mu_{B'}^l)], \quad \forall l' \quad (5-129)$$

$$L(l') = \operatorname{argmax}_{l=1 \sim R(\nu-1)} [E(\mu_{B'}^{l'}, \mu_{B'}^l)], \quad \forall l' \quad (5-130)$$

则按照下列准则进行判断和转移：

若  $D(l') \geq \theta(\nu)$  且  $L(l') = l'$ , 则无需进行结构学习, 直接转向参数学习。 $\theta(\nu)$  是一个随  $\nu$  而变化的相似度阈值参数 当学习开始时 (即  $\nu$  值较小)  $\theta(\nu)$  较小; 当  $\nu$  增加时  $\theta(\nu)$  缓慢增加。此外, 令  $R(\nu) = R(\nu-1)$ 。

若  $D(l') < \theta(\nu)$  则增加一个新节点, 即新增一个输出模糊区, 即令  $R(\nu) = R(\nu-1) + 1$ 。转向“加一新节点”模块。

若  $D(l') \geq \theta(\nu)$  而  $L(l') \neq l'$  这时无需增加新节点 即令  $R(\nu) = R(\nu-1)$ 。但是需转向“改变推理规则”模块。

#### (4) 加一新节点

若  $D(l') < \theta(\nu)$  则输出空间增加一个隶属函数  $\mu_{B'}^l(y)$ ,  $l = R(\nu)$  其中心值等于  $b_0^l(\nu)$ , 其宽度系数为  $\sigma_0^l(\nu)$ 。然后转向“改变推理规则”模块。

#### (5) 改变推理规则

设原建推理规则为 (见式 5-3))

$$R^{l'}: \mu^{l'}(X) = \max[\mu_A^{i_1^{(l')}}(X), \dots, \mu_A^{i_p^{(l')}}(X)], \quad l' = 1 \sim R(\nu-1)$$

如某  $l'$  对应于加一新节点  $l = R(\nu)$  或  $L(l') \neq l'$  情况 则将相应的规则定为

$$R^l: \mu^l(X) = \max[\mu_A^{i_1^{(l)}}(X), \dots, \mu_A^{i_p^{(l)}}(X)], \quad l = R(\nu) \quad (5-131)$$

其中  $i_1(l) \sim i_p(l)$  诸标号与  $\mu_A^{i_1^{(l')}}(X(\nu-1)), \dots, \mu_A^{i_p^{(l')}}(X(\nu-1))$  之中大于某个阈值  $\beta$  者的标号一致, 即这些隶属函数值超过某一标准后, 才得作为新建规则的前提 (一般取  $\beta = 0.5 \sim 0.8$ )。若  $l = L(l')$  令  $R^{L(l')}$  的前提中包含满足上述条件的各输入隶属函数标号。 $\beta$  值由经验决定, 一般情况可产生良好效果。但是也可能出现  $\mu_A^{i_1^{(l')}}(\nu-1), \dots$  诸隶属函数均未超过  $\beta$  而其取值仍较高的情况 (通常情况是只有少数几个隶属函数值特别高并且超出了  $\beta$ ), 对于这种例外情况不能将相应的各个前提从  $R^l$  中去除, 而应将  $\beta$  值做自适应调低<sup>[47]</sup>。

#### (6) 参数学习

在结构调整完成后, 可用 BP 算法对网络中的各隶属函数参数予以调整, 计算公式可参考文献 [45]。

#### (7) 规则合并

当全部训练完成后, 应对网络中各神经元进行检查, 判断是否存在重复冗余的情况并予以剪除合并<sup>[51]</sup>。

### 5. 应用举例 —— 倒摆平衡问题 (inverted pendulum-balancing problem)

由于倒摆平衡问题是一个典型的困难控制问题, 已经成为检验各种控制算法效果的标准问题<sup>[52]</sup> 其中包括 NN(BP)<sup>[53]</sup> 和增强学习<sup>[54,48]</sup> 下面介绍此问题。见图 5-18 给出的



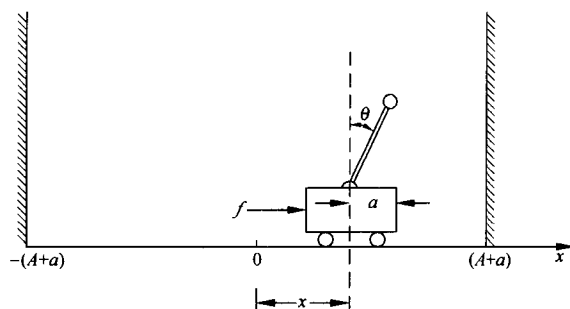


图 5-18 倒摆平衡问题的示意图

示意图，有一小车沿轨道左右行驶，车上有一摆可围绕轴心摆动，车身受力为  $f$ 。现在用状态向量  $\mathbf{X} = [x_1, x_2, x_3, x_4]$  描述此系统。其中各分量定义如次： $x_1 = x$  是车中心与轨道中心之距离，单位为米，其正、负号表示车在轨道中心的右或左侧； $x_2 = \dot{x}$  即车速，单位为米/秒； $x_3 = \theta$  单位为度，垂线右侧为正，左侧为负； $x_4 = \dot{\theta}$  单位为度/秒。 $f$  的单位为牛顿，向右为正，向左为负。车子的行动范围受左右壁限制，其与中心的距离为一  $(A+a)$ 、 $(A+a)$ 。所规定的限制是：若车行碰左或右壁，即  $|x| = A$ ，或摆角幅度超过某一最大值  $\theta_0$ ，即  $|\theta| = \theta_0$ ，则称系统产生一次失误，并产生增强信号  $r = -1$ ，反之，称系统正常，给出  $r = 0$ 。问题是：设计一个控制器，其输入为系统状态  $\mathbf{X}(\nu-1)$ ，其输出  $y(\nu-1)$  即为加于系统的力  $f(\nu-1)$ 。设计目标是系统产生两次失误之间的时间间隔应尽可能长。系统应该通过对增强信号的学习做到这一点，从而构成一个典型的增强学习问题。

根据文献[55]给出的下列典型数据可以列出系统的运行方程（可查[45]，此处不再列出），从而在计算机上进行软件模拟。重力加速度： $g = 9.8\text{m/s}^2$ ，小车加摆的质量： $m = 1.1\text{kg}$ ，摆重： $m_p = 0.1\text{kg}$ ，摆头到轴的摆长： $l = 0.5\text{m}$ ，车与轨之间摩擦系数： $\mu_c = 0.0005$ ，摆杆与轴之间摩擦系数： $\mu = 0.000002$ 。规定  $A = 2.4\text{m}$ ， $\theta_0 = 12^\circ$ ， $10\text{N}$   $f \leq 10\text{N}$ 。系统按离散时间  $\nu = 0, 1, 2, \dots$  运行，相邻离散时间之间间隔  $\Delta = 0.02\text{s}$ 。下面给出文献[45]给出的部分实验结果。模拟实验用 FP 和 FLC 两个 FNN 构成控制器，FP 采用上列 2 节（3 项给出的时差学习——情况 3 的计算公式，目标函数（即被预测值）为

$J(\nu-1) = \sum \gamma^k r(\nu+k)$ ，其中增强信号的定义是

$$r(\nu) = \begin{cases} -1, & |\theta| \geq 12^\circ \text{ 或 } |x| \geq 2.4\text{m} \\ 0, & \text{其他} \end{cases}$$

模拟实验中的初始化参数是：网络输入变量  $x, \dot{x}$  和  $\theta$  各分成 3 个均匀间隔的模糊区， $\theta$  分成 7 个均匀间隔的模糊区，相应模糊隶属函数的中心及宽度在学习前的初值见表 5-2。输出变量  $y = f$  分成 7 个模糊区，其隶属函数初值也在表 5-2 中给出。

全部模拟实验共进行 10 轮（run），每轮中包含若干次实验。每次实验从一定的初值出发，运行直至产生失误终止，然后开始下一次实验。每一轮的运行时间节拍限定为 500000 步且最多不超过 50 次实验。状态初值分为零初值（设  $x(0) = \dot{x}(0) = \theta(0) = \dot{\theta}(0) = 0$ ）和随机初值两种情况，后者在下列范围内取均匀分布： $-2 \leq x(0) \leq 2$ ， $-10 \leq \dot{x}(0) \leq 10$ ， $-10 \leq \theta(0) \leq 10$ ， $-50 \leq \dot{\theta}(0) \leq 50$ 。

表 5-2学习前后的隶属函数参数值

变量	编号	学习前		学习后	
		中心	宽度	中心	宽度
$x/\text{m}$	0	-2.4	2.6	-1.98	3.11
	1	0.0	1.5	0.13	1.28
	2	2.4	2.6	2.17	4.13
$\dot{x}/\text{m/s}$	0	-20.0	20.0	-15.42	17.21
	1	0.0	10.0	-0.21	7.39
	2	20.0	20.0	17.25	16.01
$\theta/(^{\circ})$	0	-12.0	6.0	-10.22	7.39
	1	-6.0	0.0	-5.01	4.81
	2	-3.0	3.0	-1.82	2.12
	3	0.0	3.0	0.21	1.92
	4	3.0	3.0	2.19	1.89
	5	6.0	6.0	5.31	3.87
	6	12.0	6.0	8.82	8.01
$\dot{\theta}/^{\circ}/\text{s}$	0	-100.0	100.0	-84.29	84.62
	1	0.0	50.0	-0.32	39.42
	2	100.0	100.0	73.18	110.39
$f/\text{N}$	0	-10.0	6.0	-9.48	4.21
	1	-6.6	5.0	-7.89	2.89
	2	-3.3	5.0	-6.42	2.72
	3	0.0	5.0	-4.03	3.02
	4	3.3	5.0	-2.01	1.36
	5	6.6	5.0	-1.32	0.98
	6	10.0	6.0	-0.98	0.72
	7	—	—	-0.19	0.57
	8	—	—	1.12	0.33
	9	—	—	1.86	0.68
	10	—	—	2.77	1.03
	11	—	—	4.59	2.92
	12	—	—	7.21	3.57
	13	—	—	8.88	3.24
	14	—	—	10.26	4.56

模拟实验的结果简述如下。

(1) 每一轮中开始的若干次实验由于控制器的参数尚未调至最佳, 从开始至产生失误的运行步数较少。图 5-19(a) 给出了开始阶段一次实验的示例, 取零状态初值, 运行至 450 步时由于  $|\theta| \geq 12^{\circ}$  而产生失误。对于较后的实验, 由于参数已调得较佳所以从开始至失误的步数较多。图 5-19(b) 给出这种示例, 从零初始状态出发经过约 2800 步运行, 由于  $|x| > 2\text{m}$  而产生了失误。以上实验结果取自文献[45] 在其 10 轮实验中一半取零状态初值 另一半取随机状态初值 前者的学习速度快于后者。无论哪一种初值 每一轮中最多进行 20 次试验的学习就可以学得很好, 这时被控系统几乎不会产生失误。一般进行 10 次试验就能达到很好的学习效果。

(2) FLC 在学习完成后的结构如图 5-20 所示。网络的第 2 层完成 4 个输入变量隶属函数的计算, 各隶属函数在学习后的参数值见表 5-2。由上述各变量的隶属函数相乘可以构成 189 种不同的输入空间隶属函数 ( $189 = 3 \times 3 \times 7 \times 3$ )。但是真正得到使用的只有 35

种,即  $q = 35$  网络第 3 层完成其计算。网络输出空间在学习后从学习前的 7 扩大为 15 其隶属函数的参数值见表 5-2。这说明系统规则数在学习后成为  $r = 15$  各推理规则计算在第 4 层完成。

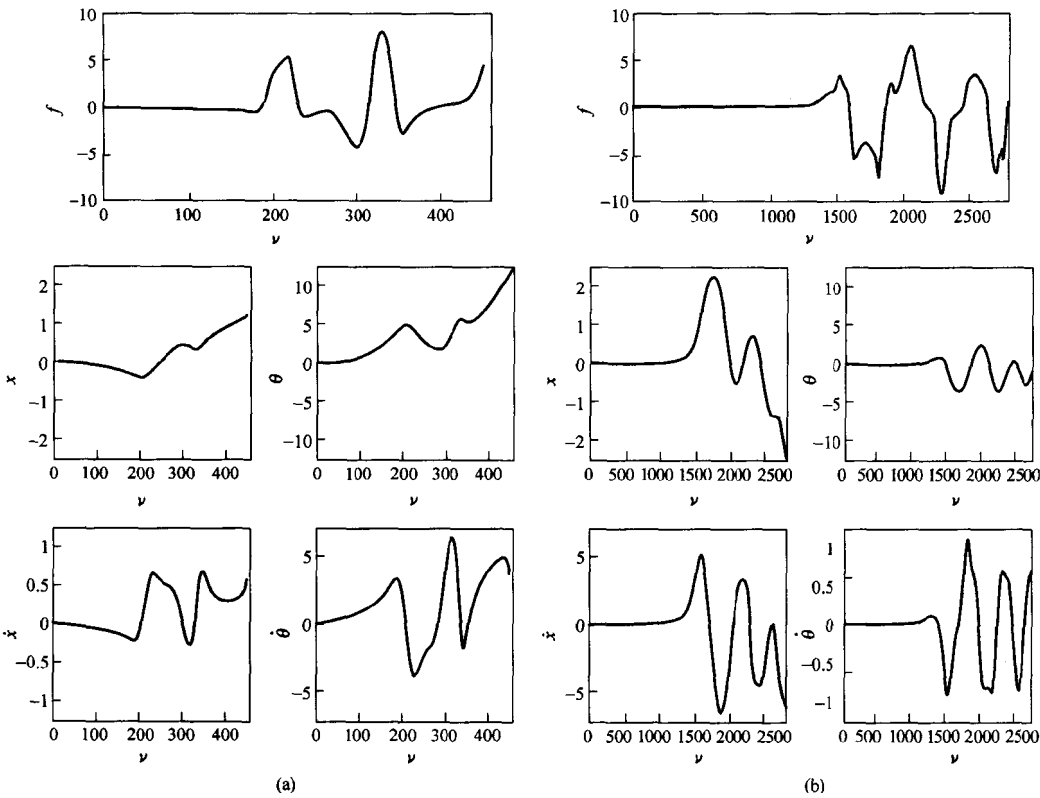


图 5-19 模拟实验结果

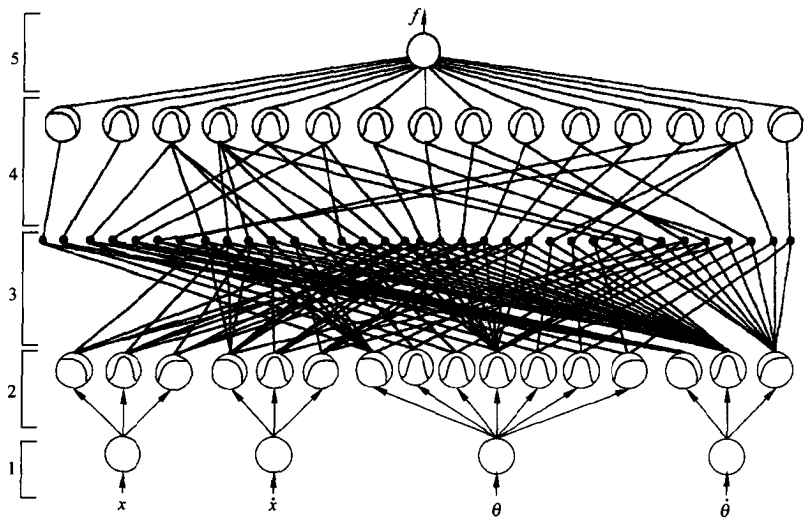


图 5-20 学习完成后的 FLC 结构图

## 5.7 FNN 用于时间序列预测及其在金融和财务等领域中的应用

对于未来的预期 (anticipation)、预测 (prediction) 或预报 (forecasting) 是人类智能的核心内容之一，根据对未来的预期进行规划、作出决策以及采取必要的行动是人类控制策略中无处不在特征，总之，人类具有预见功能是自然智能的重要标志。生物学、生态学和认知科学的最新进展以及基于计算机的预测模型研究的深入都表明，在未来的智能系统和控制系统（包括工业、经济和社会中的控制系统）中都应更加重视和强调预测和预报的应用。这一点 在上列 5.6 节关于增强学习的介绍中已予以强调。然而，对未来作出精确的预测（即便是对不远的未来进行预测）相当困难。对于只取直线或周期性运动的系统，可以用基于牛顿力学的模型进行预测。对于复杂的动力系统（高度非线性以及系统的输入和状态向量的维数很高的情况）虽然可用非线性差分方程（或微分方程）来建模，但建模精度受到多方面限制，适用面窄。特别当环境中存在噪声、干扰时困难更大。对于具有非平稳和混沌特性的系统，则更难建立适于进行预测的模型。人们凭藉经验和直觉能对未来作出一定预测，但是对这些预测往往不能作出解释，更不能揭示其中深层次的、难以简单发现的规律，从而不能充分利用积累的资料和数据。通过人工神经网络、模糊推理和遗传算法相结合，为实现预测建模提供了一个重要的方向，本小节将通过一些实例予以介绍。

预测系统的应用范围很宽，从上一节所述的人驾驶汽车的过程即需用到预测，直至复杂的工业控制、通信智能管理、机器人、Internet 的智能体 (Agent)、军事等各种领域<sup>[56]</sup>。在这一节将着重讨论预测在金融领域中的应用。随着经济的全球化和市场经济的发展，各种金融预测和判断的重要性日益增加，1998 年的亚洲金融风暴就是一个最近的例证。在这一小节中将介绍在经济、财政和金融等领域中进行预测时所涉及的若干问题和可能的解决方案，并且介绍股价和期指预测的一些实例。

### 5.7.1 金融和财政预测的内容、问题和可能的解决方案

有价证券（股票和债券）交易是金融市场活动的主要内容之一，投资者须通过正确的投资选择获益。其根据是对各种金融指数（如股指）的未来走向即涨、落或持平作出准确预测以形成决策。类似的还有外汇交易、期货交易、房地产交易等等。在财经领域中需作出各种市场需求预测、各种经济指标（如增长率、通货膨胀率）预测等。对个别公司而言，需判断其坏账率、收益率等。

金融和经济系统是一个非常复杂的大系统，它是一成百万个独立投资者和机构介入的系统，每一投资者或机构都根据自己的判断作出决策。关于金融领域中各种时间序列能否预测以及如何预测的问题，各派专家的意见差异甚大。有些研究者已辨认出其中存在混沌，更不用说系统的非线性、高维数和随机性等特点，这些学者认为准确预测非常困难。一些经济学家提出了“有效市场假设”，他们认为没有一种投资策略或预测系统能使投资者持续地获得超过市场平均回报的回报。另一派研究者认为金融市场和经济系统为研究人员提供了一个研究复杂系统的最好领域。他们认为有效市场假设并不符合实际情况，并且

提出若干实例证明，采用具有高度非线性建模能力的各类神经网络，确能改善预测效果并获取可观收益<sup>[57~67]</sup>（特别是文献[62]给出了人工神经网络在金融领域应用的详尽文献）但是正如文献[57]所指出的，金融领域中时间序列预测的研究是一项利益攸关的课题。如果一种预测算法能给出好结果，那么研究者绝不会立即将其公布于众，否则其他人按此办理会很快将回报拉低到原来的水平。这就是此领域的研究报导较少、较简单的原因。即使有所报道，也是语焉不详。下面以股指预测为例，介绍已发表文献中所用的若干策略和方法（当然很不全面）。

（1）预测的对象不应是股指的绝对数值而应是其变化趋向，例如涨幅超过 5% 等。预测应按日进行，短期预测较易进行，预测期限越长可信度越低。

（2）为预测某一种股指的未来走向，不仅要利用该股指的过去历史而且应利用与该股指有关的各项市场指数（见下文 5.7.2 小节的举例）

（3）在股指按日变化的时间序列中，包含着缓慢的随年或随季节而变化的部分。在进行预测时，可以采用 LMS 算法作线性拟合以求出这一缓变部分。然后将其从时间序列中去除，以所余变化较快部分作为预测对象。最后，在预测结果中加上缓变（线性变化）部分作为总输出。更简单而有效的方法是将股指的差分作为预测器的输入。设  $S(\nu)$  是某种股指的按日变化时间序列，则可以按下式转换为差分时间序列  $dS(\nu)$ （其中  $\lg$  是以 10 为底的对数）：

$$dS(\nu) = \lg \frac{S(\nu)}{S(\nu-1)} \quad (5-132)$$

这样，可以自动去除缓变部分。另一方面，节假日的停市在逐日的时间序列中造成空白点。为使其连贯，可以将假日前一天的股价填充之，以使其连贯。

（4）利用加短时间窗取动态平均（running average）构成多重预测器输入的方法。

在用已知股指  $S(\nu), S(\nu-1), S(\nu-2), \dots$  预测未来股指  $S(\nu+1)$  趋向时，还可以利用各种加短时间窗的股指动态平均值  $S_q(\nu), S_q(\nu-1), \dots$ 。

$$S_q(\nu) = \frac{1}{q} \sum_{i=0}^{q-1} S(\nu-i) \quad (5-133)$$

$q$  值可取为 1, 2, 3, ...（注意  $S_1(\nu) = S(\nu)$ ）可以将各  $S_q(\nu)$  的差分  $dS_q(\nu)$  作为预测器输入（见 5.7.2 小节）

（5）采用小波（wavelet）分析技术，将股指序列分解为具有不同时频分辨率的序列作为预测器输入。详见 5.7.4 小节。

（6）采用 RNN。由于递归神经网络具有记忆能力，因此研究界普遍认为用 RNN 实现的预测器具有较好的效果。在对网络进行训练时，可以用 EKF 学习算法，其效率较高且更便于实现在线自适应。详情可参见第 2 章 2.13 节和文献[62]。

（7）采用模糊分析技术和 FNN。一种办法是用本章所述的各种 FNN 设计方案来实现预测器。另一种办法是采用多网络模糊预测的方法，其中每个预测器给出各自的模糊预测结果，各个结果用 RI，权值加权后取和得到总输出结果 RI 是 reliability index 的缩写，即可靠性指数。RI 取决于各网络过去给出的均方预测误差，误差越小者 RI 值越大。详见文献[57]及下文 5.7.2 小节的介绍。

(8) 采用低风险预测方法。如果预测器给出的股指走向预测与实际一致则预报正确，如果预报股指上扬而实际下跌则为错报（false alarm）反之称为漏报。由于错报的指引会使投资者造成很大损失，所以可以在训练预测器网络时对于目标函数中的不同“误差”赋予不同的权值。对于错报赋较大权值，其他情况则较小，从而能降低错报造成的风险。详见文献[62]。

(9) 采用遗传算法、混沌分析技术并借鉴各种统计分析技术的经验和技巧。文献[63]给出了一种遗传算法与模糊分析相结合的方案。

(10) 在网络的训练中采用不同的学习速率（步幅）来形成多种预测结果（类似于(6)）。

### 5.7.2 股指预测举例

这一小节以 IBM 普通股的日变化趋向预测为例，来介绍神经网络的应用<sup>[57]</sup>。设此股的日值时间序列用  $S(\nu)$  表示， $\nu$  的单位是日。另外，设有  $I$  种市场指数  $V^i(\nu), i = 1 \sim I$ 。与  $S(\nu)$  有关。现选出 13 种有关指数，即  $I = 13$ 。在表 5-3 中列出了这 13 种指数中的 10 种，其中 5 种是道琼斯分类指数，另外 5 种是标准普尔（S&P）分类指数。

表 5-3 10 种市场指数

DJIA	道琼斯工业平均(30 个公司)
DJTA	道琼斯运输平均指数(20 个公司)
DJUA	道琼斯公用事业平均指数(15 个公司)
DJ65	道琼斯 65 综合平均指数
DJFI	道琼斯期货指数
S&P 100	标准普尔 100 种指数
S&P EI	标准普尔电子仪器
S&P SC	标准普尔半导体
S&P BE	标准普尔计算机系统
S&P500	标准普尔 500 种指数

设  $S(\nu)$  和各  $V^i(\nu)$  的短时窗动态平均为  $S_q(\nu)$  和  $V_q^i(\nu), q = 1 \sim Q$  见式(5-133))， $Q$  可取为 1, 2, 3, ... 等值。其差分为  $dS_q(\nu)$  和  $dV_q^i(\nu)$  见式(5-132))。那么，所提出的任务是已知各  $dS_q(\nu), dS_q(\nu-1), \dots$  和各  $dV_q^i(\nu), dV_q^i(\nu-1), \dots$  的条件下对  $dS_q(\nu+1)$  作出尽量准确的预测。下面介绍有关的设计考虑和模拟实验结果。

#### (1) 神经网络设计

假定取  $Q$  种短时窗动态平均值，则有  $Q \times (I + 1)$  种已知的序列可以用来预测  $dS(\nu+1)$ ，它们是  $dS_q(\nu), dS_q(\nu-1), \dots, dS_q(\nu-N)$  以及  $dV_q^i(\nu), dV_q^i(\nu-1), \dots, dV_q^i(\nu-N), q = 1 \sim Q, i = 1 \sim I$  其中  $N$  是截取的“过去历史”的长度。这样，可以用  $Q \times (I + 1)$  个 RNN 来进行预测，即用  $dS_q(\nu), \dots, dS_q(\nu-N), q = 1 \sim Q$  作为  $Q$  个网络的输入来预测  $dS(\nu+1)$ ；用  $dV_q^i(\nu), \dots, dV_q^i(\nu-N), q = 1 \sim Q$  作为另外  $Q$  个网络的输入来预测  $dV^i(\nu+1)$  等等。对于后者，尚需将对于  $dV^1(\nu+1), dV^2(\nu+1), \dots, dV^I(\nu+1)$  的预测转换为对于  $dS(\nu+1)$  的预测（见下列(2)项），最后，将  $Q \times (I + 1)$  种预测结果汇

集为单一个预测输出，汇集方法见下列（3）项。每一个预测网络用具有 2 个隐层、 $N+1$  个输入和 1 个输出的 RNN 来实现， $N=15$ 。RNN 中隐层各神经元输出延时一拍后反馈回同层各神经元（见 2.13 节）。

(2)  $dV^i(\nu+1)$  至  $dS(\nu+1)$  的转换

此转换可按下式计算：

$$dS(\nu+1) = \eta^i(\nu+1) \cdot dV^i(\nu+1), \quad i = 1 \sim I \quad (5-134)$$

其中  $\eta^i(\nu+1)$  是一个随  $\nu$  而不断更新的转换系数。更新计算公式如下：

$$\eta^i(\nu+1) = \beta_u \cdot \eta^i(\nu) + (1 - \beta_u) \frac{dS(\nu)}{dV^i(\nu)}, \quad i = 1 \sim I \quad (5-135)$$

式中  $\nu$  时刻的  $dS(\nu)$  和  $dV^i(\nu)$  是可测量的，因此  $\nu+1$  时刻的转换系数  $\eta^i(\nu+1)$  可以计算。 $\beta_u$  称为更新率， $0 < \beta_u < 1$ 。若  $\beta_u$  接近于 1 则转换系数更多地依赖于  $dS(\nu)/dV^i(\nu)$  的历史积累；反之，若  $\beta_u$  接近于 0 则其更接近于此比值的当前值。设  $\beta_u$  取  $U$  种不同值，可表为  $\beta_u$ ， $u = 1 \sim U$ ，则由同一个  $dV^i(\nu+1)$  可以转换出  $U$  种不同的  $dS(\nu+1)$  预测结果。由上列（1）和本节可知，由于输入序列不同、窗动态平均不同和更新率的不同，可以产生  $(1+IU)Q$  种不同的关于  $dS(\nu+1)$  的预测，记为  $dS_l(\nu+1)$ ， $l = 1 \sim L$ ， $L = (1+IU)Q$ 。

(3) 将  $L$  种  $dS_l(\nu+1)$  汇集为单一预测  $dS(\nu+1)$  设截至时刻  $\nu$  为止，在第  $l$  项预测值和真值之间的平均绝对误差为  $E_l(\nu)$ ，则可用下式计算出  $dS(\nu+1)$

$$dS(\nu+1) = \sum_{l=1}^L RI_l(\nu) dS_l(\nu+1) \quad (5-136)$$

其中  $RI_l(\nu)$  是第  $l$  项预测的可靠性指数，其计算公式是

$$RI_l(\nu) = C_1 \exp[-C_2 \cdot E_l(\nu)] \quad (5-137)$$

其中常数  $C_2 > 0$ ， $C_2$  越大则大误差与小误差预测之间的可靠性指数差异也越大。 $RI$  相对于  $E$  的变化曲线因  $C_2$  而异的情况如图 5-21 所示。例如，当平均绝对误差  $E$  从 0.01 升至 0.1 时，若  $C_2 = 20$  则前者的  $RI$  为 0.82 后者为 0.135，二者的可信度相差约 6 倍；若  $C_2 = 27.5$  则前者的  $RI$  为 0.76 后者为 0.064，二者可信度相差约 12 倍。系数  $C_1$  是一个规格化系数，其值应设置为使下式成立

$$\sum_{l=1}^L RI_l(\nu) = 1 \quad (5-138)$$

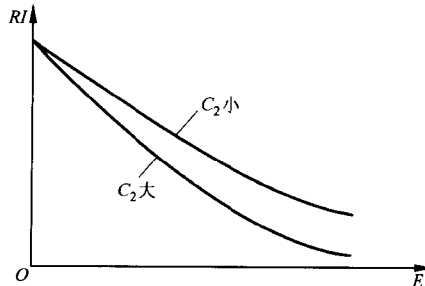


图 5-21  $C_2$  不同时  $RI \sim E$  的曲线

#### (4) 模拟实验结果

根据 IBM 普通股在 1988 至 1994 期间的股指变化曲线, 按上述策略进行投资选择。所选参数为  $I = 13, Q = 1, U = 3$ 。所得年回报率平均值为 20.9%。

### 5.7.3 期货交易举例

本例利用标准普尔 500 种指数 (S&P 500) 金融时间序列, 用神经网络和模糊处理方案, 实现回报发散度 (volatility of return) 变化的预测, 以制定高收益的期货 (option) 交易策略<sup>[57]</sup>。设  $S(\nu)$  为 S&P 500 日收盘价序列,  $\nu$  的单位是日。通过下列变换将  $S(\nu)$  转换为回报发散度序列  $x(\nu)$ 。

$$u(\nu) = \lg\left(\frac{S(\nu)}{S(\nu-1)}\right)$$

$$x(\nu) = 10^4 \left\{ \frac{1}{K+1} \sum_{k=0}^K [u(\nu-k) - \overline{u(\nu)}]^2 \right\}$$

$$\overline{u(\nu)} = \frac{1}{K+1} \sum_{k=0}^K u(\nu-k)$$

其中  $K$  是一个平均的时间框架长度。 $x(\nu)$  表现了在  $\nu-k$  至  $\nu$  的一段时间中,  $S(\nu)$  变化的剧烈程度, 即发散度。一个期货交易投资者应根据  $x(\nu)$  未来走向的预测来制定交易策略。这就是在已知当前时刻  $\nu$  及以前诸时刻的  $x(\nu), x(\nu-1), \dots$  的前提下, 预测未来时刻  $\nu+1$  的发散度  $x(\nu+1)$ , 以决定如何交易。这里不讨论期货交易的细节 (详见文献<sup>[57]</sup>) 而只讨论如何用神经网络和模糊方法尽可能准确地给出  $x(\nu+1)$  的预测。 $x(\nu+1)$  相对于  $x(\nu)$  的变化可以分成 3 种不同情况 这就是增加、持平和降低。如果用  $\tilde{x}(\nu+1) = [x(\nu+1) - x(\nu)]/x(\nu)$  来表示这一变化且其取值范围为  $-\alpha\% \sim +\alpha\%$  那么可以用 3 个模糊数: “增加  $\alpha\%$ ”、“持平”、“降低  $\alpha\%$ ” 来描述此变化并简化表示为  $r, c, l$ 。当采用三角形隶属函数时 任一  $\tilde{x}(\nu+1)$  对于此三者的隶属度值将如图 5-22 所示。例如 图中的  $\tilde{x}_a$  对  $l$  的隶属度为 0.25 对  $c$  的隶属度为 0.75 对  $r$  的隶属度为 0。预测由一个 RNN 来实现 它由一个隐层和一个输出层构成。隐层取 Sigmoid 函数 层内全反馈。输出层取线性函数 无反馈。网络的输入是  $x(\nu), x(\nu-1), \dots, x(\nu-k)$ 。网络的输出层有 3 个神经元 分别用  $l, c, r$  表示, 其输出值应分别等于  $\tilde{x}(\nu+1)$  对于相应模糊数的隶属度值。

在训练时, 应首先将训练集中的  $x(\nu+1)$  转换为  $\tilde{x}(\nu+1)$  再将其转换为对于  $l, c, r$  的隶属度值, 它们就是网络输出的理想值。

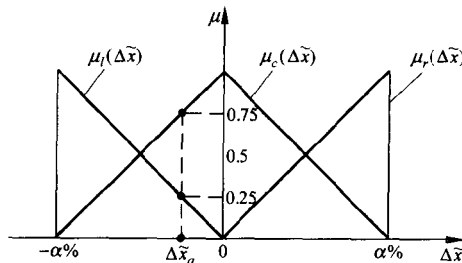


图 5-22  $\Delta\tilde{x}(\nu+1)$  对  $r, c, l$  的隶属函数值



在工作时，如果用  $\mu_l, \mu_c, \mu_r$  分别表示网络对于  $x(\nu+1)$  隶属于这 3 个模糊数隶属度的预测值，则可以分以下 8 种情况计算出网络对于  $\hat{x}(\nu+1)$  的预测值。

- ①  $\mu_l < \mu_c, \mu_l < \mu_r, \mu_c < \mu_r$
- ②  $\mu_l < \mu_c, \mu_l < \mu_r, \mu_c > \mu_r$
- ③  $\mu_l < \mu_c, \mu_l > \mu_r, \mu_c < \mu_r$
- ④  $\mu_l < \mu_c, \mu_l > \mu_r, \mu_c > \mu_r$
- ⑤  $\mu_l > \mu_c, \mu_l < \mu_r, \mu_c < \mu_r$
- ⑥  $\mu_l > \mu_c, \mu_l < \mu_r, \mu_c > \mu_r$
- ⑦  $\mu_l > \mu_c, \mu_l > \mu_r, \mu_c < \mu_r$
- ⑧  $\mu_l > \mu_c, \mu_l > \mu_r, \mu_c > \mu_r$

其中、⑥两种情况自相矛盾，实际上不可能出现。⑤、⑦两种情况不符合隶属函数特性，如出现则不进行预测。①、两种情况下应将  $\mu_l$  所做预测取消而只用  $\mu_c, \mu_r$  的预测得到总的预测结果 即

$$\Delta \hat{x}(\nu+1) = \mu_r \times \alpha\%$$

或

$$\Delta \hat{x}(\nu+1) = \mu_c \times \alpha\%$$

对于、⑧两种情况则应取消  $\mu_r$  而只用  $\mu_c$  和  $\mu_l$  得到总的预测结果，即

$$\Delta \hat{x}(\nu+1) = \mu_l \times (1 - \alpha\%)$$

或

$$\Delta \hat{x}(\nu+1) = \mu_c \times \alpha\%$$

模拟实验用了 S&P 500 在 1928 至 1993 年期间的数据，一部分用于训练，一部分用于测试。取  $K = 15, \alpha = 16.5$ 。则  $x(\nu+1)$  的预测值与实际值的部分曲线如图 5-23 所示（实线为实际值，虚线为预测值，其横轴的单位是日）。

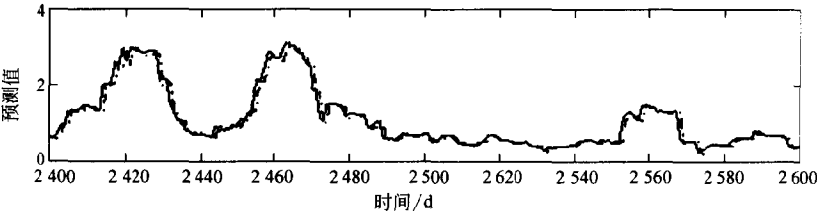


图 5-23  $x(\nu+1)$  的预测值和实际值曲线

#### 5.7.4 采用小波变换的预测器<sup>[64]</sup>

除了金融和财经领域外，时间序列的预测还用于物理科学、气象与水文的预报、管理和计划制订、军事领域等。由于时间序列的预测涉及非线性、宽频带、随机性和混沌等诸多方面问题，难度很大。这一小节着重从宽频带的角度出发，来研究预测算法的改进。用时频分析的方法，一个时间序列可以分解成频率高低不同的若干个时间序列。高频序列在时域上变化较快，在预测未来时只需用到它的较短一段过去历史。相反，低频序列在时域上变化

较慢，需用其较长的一段过去历史来预测未来。在信号处理学界已众所周知，采用傅里叶变换进行时频分析时，对于各个频段的频率分辨率和时间分辨率是固定的。而采用小波变换时，高频率段的频率分辨率低、时间分辨率高，低频率段则反之。在上述的时频分析应用中，小波变换远优于傅里叶变换。预测的任务是，在时刻  $\nu$  已知一个时间序列过去取值  $\dots, x(\nu-2), x(\nu-1), x(\nu)$  求该序列在某个未来时刻  $\nu+l$  的值  $x(\nu+l)$  ( $l$  可取  $1, 2, \dots$ )。

现在，用小波变换的方法将时间序列  $x(\nu)$  变换为不同频率段的时间序列  $\hat{x}_1(\nu), \hat{x}_2(\nu), \dots$  除了用  $x(\nu)$  来预测  $x(\nu+l)$  外，其他序列也可以用来预测，然后将各个预测值组合为单一的预测结果。下面首先讨论如何进行小波变换。

### 1. 小波变换预测器的构成

预测器结构如图 5-24 所示，图的左侧部分实施对  $x_0(\nu) = x(\nu)$  的小波变换。其中  $H_0, H_1, H_2, \dots$  为低通滤波器，依次产生  $x_1(\nu), x_2(\nu), x_3(\nu), \dots$  其中  $G_0, G_1, G_2, \dots$  为高通滤波器，依次产生  $\hat{x}_1(\nu), \hat{x}_2(\nu), \hat{x}_3(\nu), \dots$ 。设原时间序列  $x(\nu)$  的频谱在  $0 \sim f_0$  的范围内 则  $x_1(\nu)$  的频谱应在  $0 \sim f_0/2$  范围内,  $x_2(\nu)$  的频谱应在  $0 \sim f_0/4$  范围内，依次类推。 $\hat{x}_1(\nu)$  的频谱应在  $f_0/2 \sim f_0$  范围内  $\hat{x}_2(\nu)$  的频谱应在  $f_0/4 \sim f_0/2$  范围内 依次类推。由于  $x_1(\nu)$  的频带宽度较  $x_0(\nu)$  减小一半，所以其采样率也可以减少一半，即只取  $0, 2, 4, 6, \dots$  这些偶数点采样值就足够了。 $x_2(\nu)$  的频带宽度又较  $x_1(\nu)$  减小一半 其采样率又可减一半 即只取  $0, 4, 8, \dots$  采样点就够了。依此可类推至  $x_3(\nu), x_4(\nu), \dots$ 。 $\hat{x}_1(\nu)$  的采样点与  $x_0(\nu)$  相同,  $\hat{x}_2(\nu)$  的采样点与  $x_1(\nu)$  相同 余类推。以上内容可参见图 5-25。

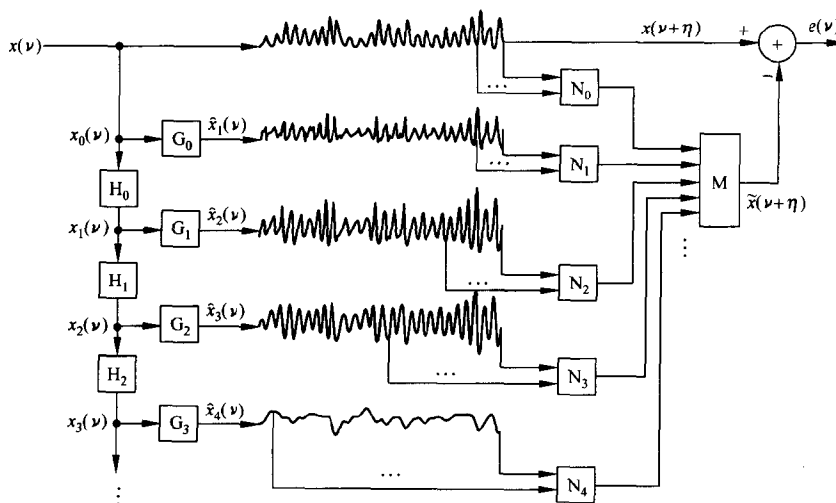


图 5-24 预测器结构图

上述变换可以采用文献 [68] 给出的快速小波变换来实现。首先定义一对函数  $\varphi(\nu)$  和  $\psi(\nu), \psi(\nu) = d\varphi(\nu)/d\nu, \varphi(\nu)$  是一个积分为 1 的对称偶函数且当  $|\nu| \rightarrow \infty$  时  $\varphi(\nu) \rightarrow 0$ 。显然  $\psi(\nu)$  的积分等于 0。设  $\varphi_l(\nu)$  和  $\psi_l(\nu)$  分别是尺度因子 (scaling factor) 为  $2^{l+1}$  时  $\varphi(\nu)$  和  $\psi(\nu)$  的扩张 (dilation) 函数 其定义为

$$\left. \begin{aligned} \varphi_l(\nu) &= \varphi(2^{-l-1}\nu) \\ \psi_l(\nu) &= 2^{-l-1}\psi(2^{-l-1}\nu), l=0,1,2,\dots \end{aligned} \right\} \quad (5-139)$$

$\varphi_l(\nu)$  称为平滑函数  $\psi_l(\nu)$  称为小波函数。当  $l=0,1,2,3$  时的  $\psi_l(\nu)$  如图 5-26 所示。

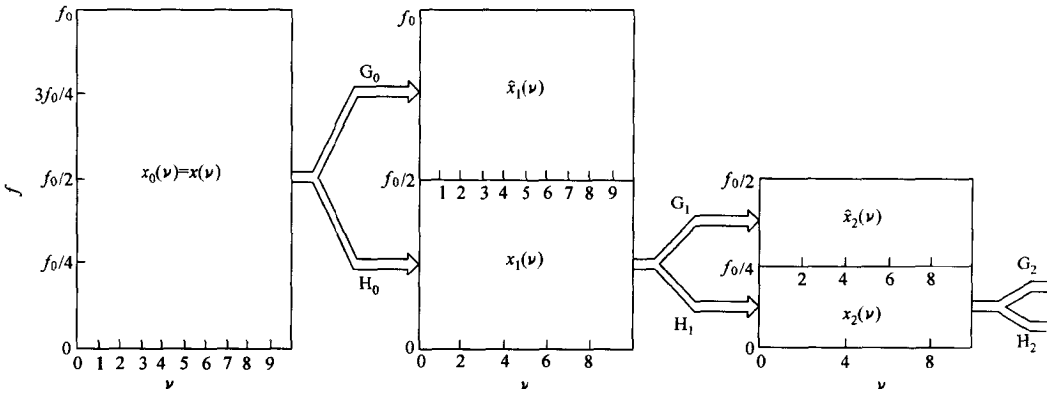


图 5-25 频率变换示意图

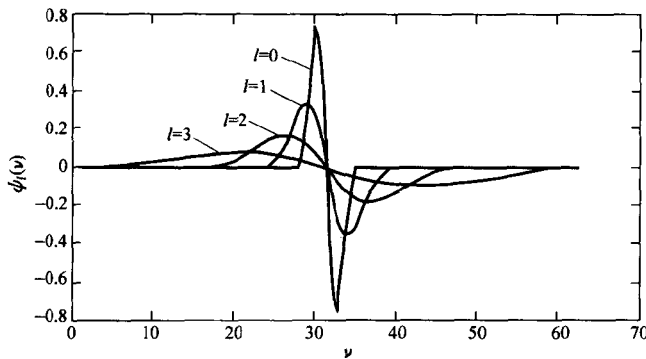


图 5-26  $l=0,1,2,3$  时的  $\psi_l(\nu)$

如果图 5-24 及图 5-25 中所示的小波变换为  $L-1$  阶的, 则可用下式计算诸  $x_l(\nu)$  和  $\hat{x}_l(\nu)$  (符号  $*$  表示求卷积):

$$\left. \begin{aligned} x_0(\nu) &= x(\nu) \\ H_l: x_{l+1}(\nu) &= x_l(\nu) * \varphi_l(\nu), \quad l=0,1,\dots,L-1 \\ G_l: \hat{x}_{l+1}(\nu) &= x_l(\nu) * \psi_l(\nu), \quad l=0,1,\dots,L-1 \end{aligned} \right\} \quad (7-140)$$

在系统中包含  $L+1$  个神经网络  $N_0, N_1, \dots, N_L$ , 它们可取各种形式的 MLFN 或 RNN。 $N_0$  的输入是  $x_0(\nu), x_0(\nu-1), \dots, x_0(\nu-\theta)$ ,  $\theta$  是一个表示过去历史长度的正整数。 $N_1$  的输入是  $\hat{x}_1(\nu), \hat{x}_1(\nu-1), \dots, \hat{x}_1(\nu-\theta)$ 。 $N_2$  的输入是  $\hat{x}_2(\nu), \hat{x}_2(\nu-2), \dots, \hat{x}_2(\nu-\theta \cdot 2)$ 。依次类推,  $N_L$  的输入是  $\hat{x}_L(\nu), \hat{x}_L(\nu-2^{L-1}), \dots, \hat{x}_L(\nu-\theta \cdot 2^{L-1})$ 。可以看到 随着  $l$  的增大,  $\hat{x}_l(\nu)$  的变化速率越慢, 相应的采样率越低且截取的过去历史越长。这正反映了小波变换的特点。而每个神经网络的输入都是  $\theta+1$  维的。这  $L+1$  个网络各自给出关于  $x(\nu+l)$  的预测 这些预测值乘以适当权值, 然后在聚合网络  $M$  中线性相加后得到最终的预测输出。这些权

值可通过 BP 算法确定 也可以按 5.7.3 节中所用的方法, 用  $RI$  系数作为权值。

## 2. 网络训练的初始权值的设置

讨论当各神经网络  $N_0 \sim N_L$  为 MLFN 时, 网络训练的初始权值该如何设置。若每一个神经网络取单隐层结构, 隐层中含  $H$  个神经元, 每个神经元取 Sigmoid 函数或 RBF。输出层只有一个神经元, 取线性函数。输入向量的维数为  $D = \theta + 1$ 。由输入至隐层第  $i$  神经元的各个权构成  $D$  维权向量  $\mathbf{W}_i^{(1)}$  (该神经元的阈值未包括在内),  $i = 1 \sim H$ 。由隐层至输出神经元的各个权构成  $H$  维权向量  $\mathbf{W}^{(2)}$  (阈值也未包括) 这些权的初值设置对于系统的预测效果和学习速度都有很大影响。文献 [64] 提出用 MLD(mean local density) 算法来确定初值。在对任一网络训练时, 将训练集中的  $D$  个输入和 1 个输出构成一个  $D+1$  维向量  $\mathbf{P}$ , 假设训练集中共包含  $K$  个  $\mathbf{P}$  向量。第一步 计算每一个  $\mathbf{P}$  向量与其他  $\mathbf{P}$  向量之间的欧氏距离或加权欧氏距离, 所谓加权是指对  $\mathbf{P}$  的最后一维 (即输出分量维) 赋予特别加大的权重。第二步 对于每一个  $\mathbf{P}$  求得与之距离最小的共  $(K/H) \cdot \rho - 1$  个向量 ( $\rho$  略小于 1 或等于 1), 并且求出它与这些最邻近向量之间的欧氏距离平均值。第三步, 以此平均值最小的  $\mathbf{P}$  及其最邻近的  $(K/H)\rho - 1$  个向量构成一个集团 求出其“质心” (一个  $D+1$  维向量), 令  $\mathbf{W}_i^{(1)}$  为此质心的前  $D$  个分量 令  $\mathbf{W}^{(2)}$  的第 1 个分量为此质心的第  $D+1$  个分量。第四步, 将此集团中所有向量从训练集中去除后回到第二步, 以求得  $\mathbf{W}_2^{(1)}$  和  $\mathbf{W}^{(2)}$  的第 2 个分量。依此类推, 直至求得  $\mathbf{W}_H^{(1)}$  和  $\mathbf{W}^{(2)}$  的第  $H$  个分量为止。

上述计算实际上是将训练集划分为  $H$  个聚类区, 每个聚类区质心的前  $D$  维决定  $\mathbf{W}_i^{(1)}$  的初值 第  $D+1$  维决定  $\mathbf{W}^{(2)}$  的相应分量初值。在训练中所有阈值分量的初值都设置为 0。

本章介绍的 FNN 主要是基于 TSK 模型构成的, 其优势在于系统框架简洁明确、便于实现以及实用效果好。近年来关于 ANN 和 FIS 如何结合的探讨很多, 二者可有多种结合方式 可参考文献 [70]。在应用方面 本章主要介绍了 FNN 在非线性动力系统辨识、控制以及时间序列预测方面的应用。实际上 FNN 还有其他许多很重要的应用领域, 例如数据采掘 [71]、信号处理 [72]、智能机器人 [69] 以及知识表示和知识发现 [73] 等。在 CI 研究中, ANN 与 FIS 的结合及 FNN 的研究是一个非常活跃的领域。除了上面列出的文献外, 读者还可以发现更多的研究课题和实际应用例证。由于篇幅限制, 本章不能一一阐述。

## 参考文献

- [1] Takagi T, Sugeno M. Fuzzy identification of systems and it's application to modeling and control. IEEE Trans. Syst. , Man, Cybern. , 1985, SMC-15:116 ~ 132
- [2] Sugeno M, Kang G T. Structure identification of fuzzy model. Fuzzy Sets Syst. , 1986, 28: 329 ~ 349
- [3] Wang L X, Mendel J M. Fuzzy basis functions, universal approximation, and orthogнал least square learning. IEEE Trans. Neural Networks, Sep. 1992, 3: 807 ~ 813
- [4] Kosko B. Fuzzy systems as universal approximators. IEEE Int. Conf. Fuzzy Syst. , 1992: 1153 ~ 1162
- [5] Ruspini E H. A new approach to clustering. Inform. Contr. , July 1969, 15(1): 22 ~ 32
- [6] Bezdek J C. Pattern Recognition with Fuzzy Objective Function Algorithms. New York: Plenum,1981
- [7] Keller J M, et al. A fuzzy k-nearest neighbour algorithm. IEEE Trans. Syst. , Man, Cybern. , 1985, SMC-15(4): 580 ~ 585
- [8] Zeng X-J, Singh M G. Approximation accuracy analysis of fuzzy systems as function approximators. IEEE Trans. on Fuzzy Systems, Feb. 1996, 4(1): 44 ~ 63
- [9] Homaifor A, McCormick E. Simutaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. IEEE Trans. on Fuzzy Systems, 1995, 3(2): 129 ~ 139
- [10] Sun C T, Jang J S. A neuro-fuzzy classifier and its applications. In: Proc. IEEE Int. Conf. Fuzzy Syst. . San Francisco(CA);Mar. 1993. I: 94 ~ 98
- [11] Juang C F, Lin C T. An on-line self-constructing neural fuzzy inference network and its applications. IEEE Trans. on Fuzzy Systems, Feb. 1998,6(1): 12 ~ 32
- [12] Barada S, Singh H. Generating optimal adaptive fuzzy-neural models of dynamical systems with applications to control. IEEE Trans. on Systems, Man, and Cybernetics-Part C: Applications and Reviews, Aug. 1998, 28(3): 371 ~ 391
- [13] Chin S L. Fuzzy model identification based on cluster estimation. J. Intell. Fuzzy Syst. . 1994, 2: 267 ~ 278
- [14] Yager R R, Filev D P. Generation of fuzzy rules by mountain clustering. J. Intell. Fuzzy Syst. , 1994, 2: 209 ~ 219
- [15] Sun C T. Rule-base structure identification in an adaptive-network-based fuzzy inference system. IEEE Trans. on Fuzzy Systems, Feb. , 1994, 2(1): 64 ~ 74
- [16] Lin Y, et al. Using fuzzy partitions to create fuzzy systems from input-output data and set the initial weights in a fuzzy neural Network. IEEE Trans. on Fuzzy Systems, Nov. 1997, 5(4): 614 ~ 621
- [17] Lin Y, Cunningham G A III. A new approach to fuzzy-neural system modeling. IEEE Trans. on Fuzzy Systems, May 1995, 5(2): 190 ~ 198
- [18] Wang L, Langari R. Building sugeno-type models using fuzzy discretization and orthogonal parameter estimation technigues. IEEE Trans. on Fuzzy Systems, Nov. 1995, 3(4): 454 ~ 459
- [19] Wang L, Mendel J. Fuzzy basis functions, universal approximation and orthogonal least-squares learning.

IEEE Trans. on Neural Networks, Sep. 1992, 3(5): 807 ~ 814

- [20] Wang L, Langari R. Complex Systems modeling via fuzzy logic. IEEE Trans. on Systems, Man, and Cybernetic-PARTB: Cybernetics, Feb. 1996, 26(1): 100 ~ 106
- [21] Jang Jyh-Shing Roger, ANFIS: adaptive-network-based fuzzy inference system. IEEE Trans. on Systems, Man, and Cybernetics, 1993, 23(3): 665 ~ 683
- [22] Carpenter G A, et al. Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. Neural Networks, 1991, 4: 759 ~ 771
- [23] Carpenter G A, et al. Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. IEEE Trans. Neural Networks, May 1992, 3: 698 ~ 712
- [24] Simpson Patrick K. Fuzzy min-max neural networks-part 1: Classification. IEEE Trans. on Neural Networks, 1992, 3(5): 776 ~ 786
- [25] Simpson P K. Fuzzy min-max neural networks-Part 2: Clustering. IEEE Trans. on Fuzzy Systems, 1993, 1(1): 32 ~ 45
- [26] Lin C J, Lin C T, An ART-based fuzzy adaptive learning control network. IEEE Trans. on Fuzzy Systems, Nov. 1997, 5(4): 477 ~ 496
- [27] Frank T, et al. Comparative analysis of fuzzy ART and ART-2A network clustering performance. IEEE Trans. on Neural Networks, May, 1998, 9(3): 544 ~ 559
- [28] Joshi A, et al. On neurobiological, neuro-fuzzy, machine learning, and statistical pattern recognition techniques. IEEE Trans. on Neural Networks, Jan. 1997, 8(1): 18 ~ 31
- [29] Tsao E C-K, et al. Fuzzy kohonen clustering networks. Pattern Recognition, 1994, 27(5): 757 ~ 764
- [30] Kohonen T. Self-organization and associative memory. 3rd ed. Berlin: Springer, 1989
- [31] Huntsberger T, et al. Parallel self-organizing feature maps for unsupervised pattern recognition. Int. J. Gen. Syst., 1989, 16: 357 ~ 372
- [32] Pal N R, et al. Generalized clustering networks and Kohonen's self-organizing scheme. IEEE Trans. on Neural Networks, July 1993, 4(4): 549 ~ 557
- [33] Karayiannis N B, et al. Repairs to GLVQ: A new family of competitive learning schemes. IEEE Trans. on Neural Networks, Sep. 1996, 7(5): 1062 ~ 1071
- [34] Karayiannis N B, Pai P.-I. Fuzzy algorithms for learning vector quantization. IEEE Trans. on Neural Networks, Sep. 1996, 7(5): 1196 ~ 1211
- [35] Gonzalez A I, et al. An analysis of the GLVQ algorithm. IEEE Trans. on Neural Networks, July 1996, 6(4): 1012 ~ 1016
- [36] Tanaka K, et al. Modeling and control of carbon monoxide concentration using a neuro-fuzzy technique. IEEE Trans. on Fuzzy Systems, Aug., 1995, 3(3): 271 ~ 279
- [37] Ivakhnenks A G, et al. Principle versions of the minimum bias criterion for a model and an investigation of their noise immunity. Soviet Automat. Contr. 1978, 11: 27 ~ 45
- [38] Nie J, Linkens D A. Learning control using fuzzified self-organizing radial basis function network. IEEE Trans. on Fuzzy Systems, Nov. 1993, 1(4): 280 ~ 287
- [39] Box G E P, Jenkins G M. Time series analysis: forecasting and control. 2nd ed. San Francisco: Holden-Day, 1976
- [40] Sugeno M, Tanaka K. Successive identification of a fuzzy model and its applications to prediction of a complex system. Fuzzy Sets Syst., 1994, 42: 315 ~ 334

- [41] Sugens M, Yasukawa T. A fuzzy-logic-based approach to qualitative modeling. IEEE Trans. Fuzzy Syst. , Feb. 1993, 1: 7 ~ 31
- [42] Narendra K S, Parthasarathy K. Identification and control of dynamical systems using neural networks. IEEE Trans. on Neural Networks, Jan. 1990, 1(1): 4 ~ 27
- [43] Wang L X, Adaptive fuzzy systems and control: design and stability analysis. Englewood cliffs, NJ; Prentice-Hall, 1994
- [44] Park Y-M, et al. A self-organizing fuzzy logic controller for daynamic systems using a fuzzy auto-regressive moving average (FARMA) model. IEEE Trans. on Fuzzy Systems, Feb. 1995, 3(1);75 ~ 82
- [45] Lin C-T, Lee C S G. Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems. IEEE Trans. on Fuzzy systems, Feb. 1994, 2(1): 46 ~ 63
- [46] Dickinson A. Contemporary animal learning theory. Cambridge MA; Cambridge University Press, 1980
- [47] Lin C-J, Lin C-T. Reinforcement learning for an ART-based fuzzy adaptive learning control network. IEEE Trans. on Neural Networks, May 1996, 7(3): 709 ~ 731
- [48] Bavto A G, et al. Neurallike adaptive elements that can solve difficult learning control problem. IEEE Trans. on Syst. , Man, Cybern. , 1983, SMC-13(5): 834 ~ 847
- [49] Sutton R S. Learning to predict by temporal difference. Machine Learning, 1988, 3: 9 ~ 44
- [50] Werbos P J. A menu of design for reinforcement learning over time. In: Miller W T, III ,et al, ed. Neural Network for control, ch. 3. Cambridge, MA; MIT Press, 1990
- [51] Lin C T, Lee C S G. Neural-network-based fuzzy logic control and dicision system. IEEE Trans. Comput. , Dec. 1991 C-40(12): 1320 ~ 1336
- [52] Cheok K C, Loh N K. A ball-balancing demonstration of optimal and disturbance-accommodating control. IEEE Contr. Syst. Mag. , Feb. 1987; 54 ~ 57
- [53] Widrow B. The original adaptive neural net broom-balancer. In; Proc. Int. Symp. Circ. and Syst. May 1987. 351 ~ 357
- [54] Anderson C W. Strategy learning with multilayer connectionist representations. In; Proc. Fourth Int. Workshop on Mach. Learn. . Irvine, CA; June 1987. 103 ~ 114
- [55] Anderson C W, Miller W T III. Challenging control problems. In; Miller W T III. Sutton R S and Werbos P J, ed. Neural Networks for Control. Cambridge, MA; MIT Press, 1990
- [56] Tsoukalas L H. Neurofuzzy approaches to anticipation; a new paradigm for intelligent systems. IEEE Trans. on SMAC-part B; Cybernetics, Aug. 1998, 28(4): 573 ~ 582
- [57] Pantazopoulos K N, et al. Financial prediction and trading strategies using neurofuzzy approaches. IEEE Trans. on SMAC-Part B; Cybernetics, Aug. 1998, 28(4);520 ~ 531
- [58] Refenes P, et al, Eds.. Neural networks in financial engineering. Singapore; World Scientific, 1996
- [59] Trippi R, Lee K. Artificial intelligence in finance & investing. Chicago, IL, Irwin; 1996
- [60] Tsoukalas L H, Uhrig R E. Fuzzy and neural approaches in engineering. New York; Wiley, 1997
- [61] Hobbs A; Bourbakis N G. A neurofuzzy arbitrage simulator for stock investing. In; Proc. IEEE/IAFE 1995 Conf. Computational Intelligence Financial Engineering. New York; Apr. 9 ~ 11, 1995
- [62] Saad Emad W, et al. Comparative study of stock trend prediction using time delay, recurrent and

- probabilistic neural networks. *IEEE Trans. on Neural Networks*, 1998, 9(6) : 1456 ~ 1468
- [63] Kim Daijin, Kim Chulhyun. Forecasting Time series with genetic fuzzy predictor esemble. *IEEE Trans. on Fuzzy Systems*, Nov. 1997 : 5(4): 523 ~ 535
- [64] Geva Amir B. ScaleNet—multiscale neural network architecture for time series prediction. *IEEE Trans. on Neural Networks*, Nov. 1998, 9(6): 1471 - 1482
- [65] Kwon T W, Feroz E H. A multilayer perceptron approach to prediction of the SEC'S investigation targets. *IEEE Trans. on Neural Networks*, Sep. 1996. 7(5): 1286 ~ 1290
- [66] Donaldson R G, Kamstra M. Forecasting combining with neural networks. *J. Forecasting*, 1996. 15(1): 49 ~ 61
- [67] 曾勇. 人工神经网络在资本市场预测中的应用: [清华大学内部报告]. 1997
- [68] Mallat S G, Zhong S. Characteristic of signal from multiscale edges. *IEEE Trans. Pattern Anal. Machine Intell.* 1992. 10: 710 ~ 732
- [69] Fukuda T, Kubota N. An intelligent Robotic system based on fuzzy approach. *Proceedings of the IEEE*, Sep. 1999, 87(9) : 1448 ~ 1470
- [70] Kung S Y, et al. Synergistic modeling and applications of hierarchical fuzzy neural networks. *Proceedings of IEEE*, Sep. 1999. 87(9): 1550 ~ 1574
- [71] Hirota K, Pedrycz W. Fuzzy computing for data mining. *Proceedings of the IEEE*, Sep. 1999 : 87(9): 1575 - 1600
- [72] Plataniotis K Y, et al. Adaptive fuzzy systems for multichannel signal processing. *Proceedings of the IEEE*, Sep. 1999. 87(9): 1601 - 1622
- [73] Giles C L, et al. Equivalence in knowledge representation: automata, recurrent neural networks, and dynamical fuzzy systems. *Proceedings of the IEEE*, Sep. 1999, 87(9) : 1623 ~ 1640
- [74] 李松银 郑君里. 前向多层神经网络模糊自适应算法. *电子学报*, 1995. 2, 23(2): 1 ~ 6
- [75] 郑佩, 郑君里. 用模糊神经网络求解多级交换网路由重排问题. *清华大学学报*, 1997. 1, 37(1) : 18 ~ 21
- [76] Li Songyin, Zheng Junli The fuzzy adaptive algorithm for MLFN. *CJE(Chinese Journal of Electronics)*. 1994. 10, 3(4): 28 - 33



# 第 6 章 遗传算法及其在人工神经网络中的应用

## 6.1 概 述

虽然人工神经网络的应用日益广泛，但是仅仅依靠神经网络自身构成的各种算法往往未能显示其突出的优点。而将人工神经网络与其他计算智能算法相互结合有可能产生很好的应用效果。在第 5 章已经看到了神经网络与模糊集理论相互结合取得的研究成果及其发展动向。本章将要介绍当代计算智能的另一研究领域——遗传算法，并给出遗传算法与神经网络相互结合的算法构成原理与应用实例。

人工神经网络的主要应用领域之一是求解优化问题，而遗传算法也是一种求解优化的有效方法。

优化问题寻找最优解所采取的途径多种多样。这些途径也可称为搜索方法。最基本的搜索方法是枚举算法。这种算法原理简单，而且从理论上讲可以求得待解决问题的最优解。然而，它的计算时间、占用空间可能规模很大，在许多实际问题中将遇到很大困难。针对这种最优化算法之不足，人们提出了启发式算法，它的构成原理是基于直观或经验，使搜索过程有一定指导性，在可接受的花费（时间与空间）之下给出待解决问题的可行解，此可行解与最优解之间可能存在偏差。

启发式算法的研究源远流长，多种启发式算法已经形成内容丰富的优化研究领域。而从 20 世纪 80 年代开始，逐步兴起的一些方法被称为现代优化算法。在这些算法中，模拟退火优化方法是利用热力学的进化过程寻找能量最小状态，而神经网络方法则是建立能量函数方程经求解找到最优或次优结果。20 世纪后期，利用进化算法的优化方法引起人们的巨大兴趣，应用日益广泛。它的原理是根据自然界的生物进化理论，通过模仿生物的遗传过程在代与代之间传递最优解的信息，完成搜索任务。这类方法又可进一步划分为遗传算法（genetic algorithm, GA）、进化策略（evolutionary strategy, ES）、进化规划（evolutionary programming, EP）等 3 种方法。它们的共同特点是可以对优化问题进行全局搜索，算法性能具有很好的鲁棒性，算法实现具有明显的灵活性。限于篇幅，本书只讨论遗传算法。

按照达尔文主义的进化理论，生命历程包括繁殖、变异、竞争和选择等多种复杂过程。在繁殖过程中，个体的遗传物质将复制给后代，所谓遗传物质是指染色体，它的基本单元称为基因。个体所包含的基因决定了个体的特性。在信息传递过程中基因复制可以发生错误，这就产生了所谓变异。变异过程的存在使得生物繁衍具有遗传与变异的双重特性，使生物群落既有一定的稳定性又有一定的可变性与多样性。竞争是在有限资源空间中生物

个体数量持续扩张产生的，而选择则是在物种充满整个可获得空间后相互竞争的必然结果。这些相互作用的随机运行构成了生命的进化过程。遗传算法就是受到生命进化原理的启示而形成的一种寻优方法。

从 20 世纪 60 年代开始，美国密歇根大学的 J. Holland 教授对这种模仿自然进化系统而设计人工优化系统的方法进行了广泛深入的研究。1975 年他的专著《自然系统和人工系统的适配》出版。并提出了对遗传算法理论研究具有重要意义的模式理论。此后，众多学者加入这一研究领域，并逐步形成研究热潮。到 90 年代，遗传算法已在许多科学与技术领域得到广泛应用 例如计算机科学、自动控制、机器人学、模式识别、人工神经网络等。

遗传算法的基本流程如图 6-1 所示。首先 对可行域中的点进行编码；然后，在可行域中随机挑选一组编码（染色体、个体）作为进化起点的第一代群体，并计算每个编码的个体适应度值，而适应度体现了目标函数的寻优信息。接下来与自然界一样，从群体中随机挑选若干个体作为繁殖过程前的样本集合，选择机制应保证适应度较高的个体能保留较多的样本，而适应度低的个体则保留较少样本或被淘汰。在繁殖过程中，利用交叉和变异两种算子，以一定的交叉率和变异率对挑选后的样本进行变换，从而给出新个体。最后，通过新老个体替换产生下一代群体。算法不断重复进行上述评价、选择、繁殖和替换过程，直到结束条件得到满足为止。通常，进化过程最后一代群体中适应度最高的个体就是利用遗传算法求解最优化问题的最终结果。

从以上简要介绍可以看出，遗传算法最主要的特点是：利用适应度提供的信息进行搜索，不需要其他的辅助信息；选择、交叉和变异这三个算子都是随机操作的，不需确定规则控制。

概括讲，遗传算法将适者生存原则与随机信息交换结合起来，前者力图消除解中的不适应性因素，后者继承了原有解中已有的知识，从而有力地加快了算法的搜索过程。

6.2 节将给出利用遗传算法求解优化问题的实例。6.3 和 6.4 节将进一步讨论遗传算法的运行机理，包括模式定理和收敛性能分析。6.5 节介绍各种改进的遗传算法。最后，在 6.6 节讨论遗传算法与人工神经网络相结合的应用实例。

## 6.2 基本的遗传算法

函数优化问题可表述为：有一  $n$  维未知函数  $f(x); R^n \rightarrow R$  当输入一自变量时 能得知相应的函数值，据此求  $\max f(x)$ 。这是一个黑箱问题，没有函数在连续性等方面的任何信息，我们期望借助某种搜索算法找到  $f(x)$  的最大值。好的搜索算法必须具有较好的鲁

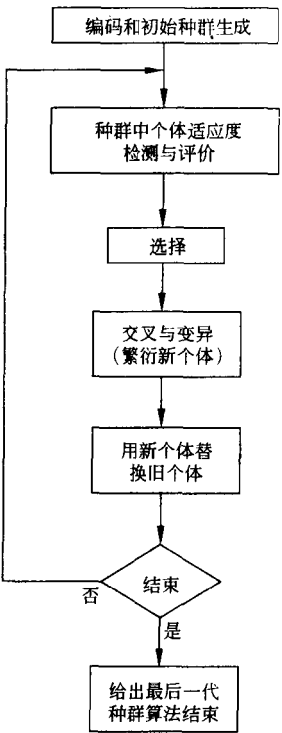


图 6-1 遗传算法的基本流程

棒性以及较高的效率。

遗传算法把该问题中的自变量当作生物体，将其转化为由基因构成的染色体，相应的函数值定义为适应度，未知函数为环境，问题的目标变为求进化成具有最佳适应度的基因型。把染色体称为串，用遗传算法求解上述函数优化问题的步骤如下：

- (1) 选择编码策略，把参数转换成串；
- (2) 选择群体大小  $N$  随机产生  $N$  个串构成群体；
- (3) 根据适应度函数（目标或耗费函数） $F = f(x)$  计算各个串的适应度；
- (4) 根据串的复制概率  $p_r(f)$  选择一个串进行复制，直到已经复制了  $N$  个串 适应度越高，复制概率越大；
- (5) 复制后的串两两配对，以交叉概率  $p_c$  进行交叉；
- (6) 对每个串中的基因按变异概率  $p_m$  进行翻转；
- (7) 从(3)起重复进行，直到满足某一性能指标或规定的遗传代数。

以上遗传过程描述了最简单的进化模型。(4)至(6)进行3种基本基因操作，复制实施适者生存的原则；交叉的作用是组合父代中有价值的信息，产生新的后代，以实现高效搜索；变异的作用是保持群体中基因的多样性。

例 6-1 假定用遗传算法求函数  $f(x) = x$  的最大值  $x \in [0, 255]$ 。表 6-1 给出了用遗传算法求解此问题的计算过程和结果。

下面结合表 6-1，对上述遗传算法的各个步骤作具体说明。

(1) 编码

由于遗传算法不能直接处理空间的数据，因此我们必须通过编码将它们表示成遗传空间的基因型串结构数据。在表 6-1 中，我们对自变量  $x$  采用 8 位二进制编码，例如  $x = 68$  表示成 01000100。

表 6-1 遗传算法求解函数  $f(x) = x$  最大值的流程

串编号	随机产生 初始群体	适应度 $f(x) = x$	选择 概率	实际被 选次数	选择后 新群体	交叉 对象	交叉 位置	交叉结果	变异	适应度 $f(x) = x$
0	10101111	175	0.189	0	1111   0001 <sup>①</sup>	1	4	11111100	11111101	253
1	01101000	104	0.112	0	1110   1100	0	4	11100001	11100001	225
2	11110001	241	0.260	4	111100   01	3	6	11110001	11110001	241
3	11101100	236	0.254	2	111100   01	2	6	11110001	11110001	241
4	00101100	44	0.047	1	1   1101100	5	1	11000101	11000101	197
5	01000101	69	0.074	1	0   1000101	4	1	01101100	01101000	104
6	00100100	36	0.039	0	1111   0001	7	4	11111100	11011100	220
7	00010111	23	0.025	0	0010   1100	6	4	00100001	00100001	33
适应度总和		928								1514
最大适应度		241								253

“1111 | 0000”中的“|”表示交叉位置。

(2) 产生初始群体

初始群体的每个个体都是随机产生，它们代表优化问题的一些可能解。一般来说，它

们的适应度较差，遗传算法的任务就是从这些较差的个体出发，模拟进化过程，优胜劣汰，最后找到非常优秀的个体，满足优化要求。在表 6-1 中 群体的大小为 8 即由 8 个个体组成 表中第 1 列为个体串编号，第 2 列为随机产生的初始种群。

(3) 适应度评价

个体的适应度由个体所代表的自变量的函数值决定。表 6-1 的第 3 列给出个体适应度。例如，个体 00100100 代表 36 其函数值为 36，因而该个体的适应度为 36。适应度是进行遗传算子操作的依据。

(4) 选择(复制)

选择或复制的目的是为了从当前种群中选出较好的个体，使它们有机会作为双亲繁殖后代，根据适者生存的原则，选择概率一般与个体的适应度成正比，即适应度越高的个体繁殖后代的机会越大，从而使优良特性得到遗传。在表 6-1 中，我们采用适应度比例方法(轮盘赌选择方法)进行选择(如图 6-2 所示)。首先计算种群中所有个体的适应度总和  $\sum f$ ，再计算每个个体适应度所占的比例  $f_i / \sum f$ ，并以此作为每个个体的选择概率，然后将轮盘按各个个体的选择概率进行分配，转动轮盘 8 次，就可以得到选择结果。表 6-1 的第 4 列为每个个体的选择概率，第 5 列为每个个体实际被选的次数，第 6 列为选择后产生的新种群。例如，种群中所有个体的适应度总和为 928 第 2 号个体的适应度为 241 因而其选择概率为  $241/928 = 0.260$  转动轮盘 其实际被选次数为 4 次。在第 6 列第 1、3、4、7 行(相应的串编号为 0,2,3,6)都选择了“11110001”。同理 串编号为 3 的码型被选择 2 次 而串编号为 4 和 5 的码型各被选择 1 次。由此构成第 6 列的新群体。

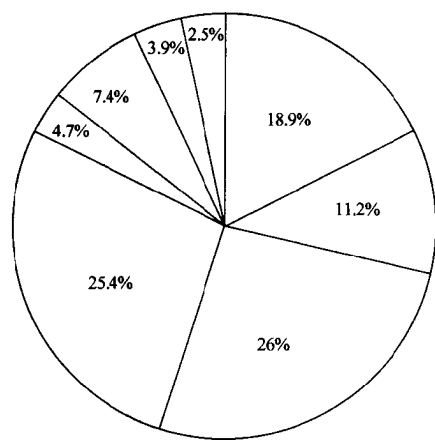


图 6-2 轮盘赌选择方法示意图

(5) 交叉

交叉算子在遗传算法中起核心作用，它把两个父代个体的部分结构加以替换重组而生成新的个体，体现了自然界中信息交换的思想。交叉概率一般较大，在表 6-1 中 我们令交叉概率为 1。表 6-1 的第 7 列给出每个个体的交叉对象，第 8 列给出交叉位置，第 9 列为交叉结果。例如 对于第 0 号个体 其交叉对象为第 1 号个体，交叉位置为 4 其实际操作过程为：

个体 0: 1111 | 0001  $\longrightarrow$  11111100

个体 1: 1110 | 1100  $\longrightarrow$  11100001

交叉位置

通过交叉，遗传算法的搜索能力得以飞跃提高。交叉过程中，既可能产生新的更好的个体成员，又可能失掉一些重要的有发展前途的个体成员，这是算法中矛盾的两个方面，类似于生物的进化和退化过程。

#### (6) 变异

变异算子是遗传算法中的一个重要算子，其作用是通过变异，补偿种群在某一位可能缺失的基因，保证遗传算法可以搜索到空间中所有点。对于二进制编码而言，变异操作就是把某些基因位上的基因值取反，即  $0 \rightarrow 1, 1 \rightarrow 0$ 。变异概率一般较小，在表 6-1 中，我们令变异概率为 0.01。表中第 10 列为变异结果，共有 3 个基因发生了变异，第 11 列为个体适应度。

以上为遗传算法一代运行的结果。由表 6-1 可以看出，经过一代的进化，种群的适应度总和由 928 上升至 1514，最大适应度由 241 上升至 253，种群获得了改善。

与传统的搜索方法相比，遗传算法具有以下特点：

(1) 遗传算法处理参数集合的编码，而不是参数本身，也就是说，其操作是在给定字符串上进行的；

(2) 遗传算法同时搜索解空间中的许多点，而不是一个点，因而能够实现快速全局收敛；

(3) 遗传算法中对个体成员的评价、产生可以并行进行，非常适合于并行算法，有利于提高算法速度；

(4) 遗传算法只需一个适应性函数（性能指标），而不需要导数或其他辅助信息，因而具有广泛的适应性；

(5) 遗传算法使用概率规则指导搜索，而不是确定性规则，因此能搜索离散的有噪声的多峰值复杂空间；

(6) 遗传算法在解空间内进行充分的搜索，但并不是盲目的穷举（评价为选择提供了依据），因此其搜索时耗和效率往往优于其他优化算法；

(7) 遗传算法运行结束时，往往是以一个群体来表示问题的解，而不是以一点来表示，因此所求得解具有鲁棒性。

## 6.3 模式定理

遗传算法模拟自然界的生物进化过程，对于解决优化问题简单而有效。但是，遗传算法的运行机制是什么？为什么通过选择、交叉、变异等基本的遗传算子操作，就可以使遗传算法具有强大的寻优能力？在本节中，我们将利用模式定理对遗传算法的运行机理进行分析。

**定义 6-1** 表示基因串中某些特征位结构相同的串，称为模式。对于二进制串，引入任意符“\*”；它既可以是“1”，也可以是“0”，那么二进制串的模式有如下形式：

$$(a_1 a_2 \cdots a_i \cdots a_l), a_i \in \{0, 1, *\}$$

属于某个模式的串，与模式在所有不是“\*”的基因位上匹配，因此模式中的“\*”越多，其表示的串就越多。

例如，对于长度为 5 的基因串，模式  $*110*$  表示位置 2、3 上为“1”且位置 4 上为“0”的基因串，即

$$*110* = \{(01100), (01101), (11100), (11101)\}$$

长度为  $l$  的二进制串共有  $3^l$  个模式，为了描述不同模式之间的差异，首先给出模式的几个重要参数的定义。

**定义 6-2** 模式  $H$  中固定位的个数称为该模式的阶，记作  $O(H)$ 。

例如  $O(*110*) = 3, O(****1) = 1$ 。模式  $H$  所包含的串的个数为  $D(H) = 2^{l-O(H)}$ ，一个模式的阶越高表示的串就越少其确定性也就越高。

**定义 6-3** 模式  $H$  中第一个固定位和最后一个固定位之间的距离称为该模式的定义长度，记作  $\delta(H)$ 。

例如， $\delta(*110*) = 2, \delta(****1) = 0$ 。

假设在遗传算法的第  $k$  代，种群中属于模式  $H$  的个体有  $m$  个，记作  $m(H, k)$ ，其适应度之总和为  $f(H)$ ，其复制概率为  $f(H)/\sum f_i$ ，因而在第  $k+1$  代，种群中属于模式  $H$  的个体的期望值为

$$m(H, k+1) = Nf(H)/\sum f_i = m(H, k) \frac{\overline{f(H)}}{\bar{f}} \quad (6-1)$$

其中  $\overline{f(H)} = \frac{f(H)}{m(H, k)}$  为模式  $H$  的个体的平均适应度， $\bar{f} = \frac{\sum_{i=1}^N f_i}{N}$  为种群中所有个体的平均适应度。可见，平均适应度高于种群平均适应度的模式将在下一代中得到增长，而平均适应度低于种群适应度的模式将在下一代中减少。

假设模式  $H$  的平均适应度高于种群平均适应度，即  $f(H) = (1+c)\bar{f}$  则

$$m(H, k+1) = (1+c)m(H, k) \quad (6-2)$$

若从  $k=0$  开始， $c$  保持恒定，则有

$$m(H, k) = m(H, 0)(1+c)^k \quad (6-3)$$

由上式可知，选择算子的作用是使平均适应度高于种群适应度的模式按指数级形式增长，使平均适应度低于种群适应度的模式按指数级形式减少。

上面我们讨论了选择算子对模式的影响。由于选择算子不能产生新的个体，即不能实现对搜索空间的全局搜索，因此必须引入交叉算子。为便于分析，我们仅讨论单点交叉对模式的影响。

显然，对于个体串所属于的某个模式来说，当交叉点落在该模式的定义长度之外时，模式不会改变，而当交叉点落在该模式的定义长度之内时，模式仍有可能不被破坏。例如：个体 11001010 属于模式  $**0****1*$ ，其定义长度为 4，当交叉点落在定义长度之外时，模式不会改变，而当交叉点落在定义长度之内，例如交叉点落在第 4 和第 5 个基因之间，若该个体的配对个体的第 7 位为 1，模式不改变，若为 0，模式改变。因此，假设交叉概率为  $p_c$ ，模式  $H$  不被交叉算子破坏的概率的下界为

$$1 - p_c \frac{\delta(H)}{l} \quad (6-4)$$

对于变异算子，假设个体串上某一位发生变异的概率为  $p_m$ ，则该位不发生变异的概率为  $1 - p_m$ 。若要模式  $H$  不被变异算子破坏，则该模式所有的固定位都不能发生变异，因而该模式保持不变的概率为  $(1 - p_m)^{O(H)}$ ， $O(H)$  为模式的阶。由于  $p_m \ll 1$  从而模式  $H$  不被变异算子破坏的概率可近似取

$$(1 - p_m)^{O(H)} = 1 - O(H) p_m \quad (6-5)$$

综上所述，我们得到如下的模式定理。

定理 6-1(模式定理) 在选择、交叉和变异三个遗传算子的作用下，定义长度短、低阶且适应度高于种群平均适应度的模式的数量在遗传过程中将以指数形式增加：

$$m(H, k+1) \geq m(H, k) \frac{f(H)}{f} \left[ 1 - p_c \frac{\delta(H)}{l} - O(H) p_m \right] \quad (6-6)$$

式中忽略了极小项  $p_c \frac{\delta(H)}{l} \cdot O(H) p_m$ 。

模式定理是遗传算法的理论基础，是指导遗传算法设计的重要原则。选择的编码策略必须使  $\delta(H)$  短、 $O(H)$  低的模式对应于所求的解。由于编码是遗传过程中的基石，不合适的编码会极大影响遗传算法的性能。

## 6.4 遗传算法的收敛性能

上一节介绍了模式定理，通过检查包含在群体中的各种模式的增长速率使我们进一步理解了遗传算法的寻优处理能力。但是，为了利用遗传算法解决优化问题，还必须保证遗传算法的收敛性，即保证遗传算法能找到问题的最优解。在文献 [4] 中 Gunter Rudolph 利用马尔可夫链的性质研究了遗传算法的收敛性。下面介绍他的证明方法。我们将注意到，前面给出的基本遗传算法不能依概率 1 收敛到全局最优值，而经过改进的遗传算法就可以依概率 1 收敛到全局最优值。

首先介绍马尔可夫链的一些基本概念。一个有限马尔可夫链描述了在一个有限状态空间  $S$  的概率转移轨迹，其中状态数为  $|S| = n$ 。在时间  $t$  从状态  $i \in S$  转移到状态  $j \in S$  的概率为  $p_{ij}(t)$ 。如果转移概率与时间  $t$  无关，则该马尔可夫链被称为同构的。

对于一个同构的马尔可夫链，其转移概率可以用一个转移矩阵  $P = (p_{ij})$  来表示。其中  $0 \leq p_{ij} \leq 1$  并且对于所有的  $i \in S$  有  $\sum_{j=1}^{|S|} p_{ij} = 1$ 。具有这种性质的矩阵被称为随机矩阵。给定一个初始分布为行向量  $p(0)$ ，经过  $t$  步转移，该马尔可夫链的分布由  $p(t) = p(0)P^t$  决定。可见，一个同构的有限马尔可夫链完全由  $(p(0), P)$  决定。马尔可夫链的极限性质取决于转移矩阵  $P$ ，下面给出关于矩阵的一些定义。

定义 6-4 一个矩阵  $A_{n \times n}$  称为：

(1) 非负的 (nonnegative) ( $A \geq 0$ ) 如果对于所有的  $1 \leq i, j \leq n$  有  $a_{ij} \geq 0$ 。

(2) 正定的 (positive) ( $A > 0$ ) 如果对于所有的  $1 \leq i, j \leq n$  有  $a_{ij} > 0$ 。

一个非负矩阵  $A_{n \times n}$  称为：

(3) 基本的 (primitive) 如果存在一个  $k$  使得  $A^k$  是正定的。

(4) 可约的 (reducible) 如果通过对行和列实施相同的变换 可以把  $A$  变换成如下形式：

$$\begin{bmatrix} C & 0 \\ R & T \end{bmatrix}$$

其中  $C$  和  $T$  为方阵。

(5) 不可约的 (irreducible) , 如果它不是可约的。

(6) 随机的 (stochastic) 如果对于所有的  $1 \leq i \leq n$ , 有  $\sum_{j=1}^n a_{ij} = 1$ 。

一个随机矩阵  $A_{n \times n}$  称为：

(7) 稳定的 (stable) , 如果矩阵具有相同的行。

(8) 列可容的 (column-allowable) , 如果矩阵的每一列都至少有一个正元素。

可以看出, 随机矩阵的乘积还是随机矩阵, 每个正定矩阵也是基本矩阵。下面给出证明遗传算法收敛性所需的几个重要定理。

**引理 6-1** 设  $C, M, S$  为随机矩阵, 其中  $M$  是正定的,  $S$  是列可容的, 则它们的乘积  $CMS$  是正定的。

**证明** 令  $A = CM, B = AS$ 。因为  $C$  是随机矩阵 则  $C$  的每一行至少存在一个正元素, 那么对于所有的  $1 \leq i, j \leq n$  有  $a_{ij} = \sum_{k=1}^n c_{ik} m_{kj} > 0$  即  $A > 0$ 。同理 因为  $S$  是列可容的, 对于所有的  $1 \leq i, j \leq n$  有  $b_{ij} = \sum_{k=1}^n a_{ik} s_{kj} > 0$  即  $B > 0$ 。证毕。

**定理 6-2** 设  $P$  为基本随机矩阵 那么当  $k \rightarrow \infty$  时  $P(k)$  收敛于一个正定稳定随机矩阵  $P(\infty) = \mathbf{1}' p(\infty)$  其中  $\mathbf{1}' = (1, 1, \dots, 1)^T, p(\infty) = p(0) \cdot \lim_{k \rightarrow \infty} P(k) = p(0) P(\infty)$  与初始分布无关, 具有非零元素并且惟一。

**定理 6-3** 设  $P$  为可约随机矩阵 其中  $C_{m \times m}$  为基本随机矩阵  $R$  和  $T$  不为  $0$  那么

$$P(\infty) = \lim_{k \rightarrow \infty} P(k) = \lim_{k \rightarrow \infty} \begin{bmatrix} C(k) & 0 \\ \sum_{i=0}^{k-1} T(i)RC(k-1) & T(k) \end{bmatrix} = \begin{bmatrix} C(\infty) & 0 \\ R_{\infty} & 0 \end{bmatrix} \quad (6-7)$$

是一个稳定随机矩阵, 其中  $P(\infty) = \mathbf{1}' p(\infty), p(\infty) = p(0) P(\infty)$  是惟一的且与初始分布无关 并且满足 对于  $1 \leq i \leq m, p_i(\infty) > 0$ ; 对于  $m < i \leq n, p_i(\infty) = 0$ 。 $R_{\infty}$  表示  $\sum_{i=0}^{k-1} T(i)RC(k-1)$  之极限。

遗传算法可以用一个马尔可夫链描述, 由于其状态由每个个体的基因决定, 因此状态空间  $S$  可表示为  $IB^N = IB^n$  其中  $n$  为种群大小,  $l$  为个体基因串长度。状态空间的每个元素可以被认为是一个用二进制表示的整数。这种映射是同构的, 状态  $i \in S$  可根据需要用二进制表示或用整数表示。投影  $\pi_k(i)$  提取状态  $i$  的二进制表示中的第  $k$  个长度为  $l$  的段, 用于区分个体和种群。

种群中的基因由于遗传算子操作而发生的概率变化可以用转移矩阵  $P$  表示 而  $P$  又可以很自然地分解为三个随机矩阵的乘积  $P = CMS$  其中  $C, M, S$  分别表示由交叉、变异和选择而引起的状态转移。

**定理 6-4** 如果变异概率  $p_m \in (0, 1)$  交叉概率  $p_c \in [0, 1]$ , 那么采用适应度比例方



法进行选择的遗传算法的转移矩阵是基本的。

证明交叉算子可以被认为是一个在状态空间  $S$  上的随机映射函数，即  $S$  的每一个状态都随机映射到另一个状态。因此，矩阵  $C$  是随机的。同理，矩阵  $M$  和  $S$  也是随机的。由于变异算子独立地作用于种群中的每一个基因，对于所有的  $i, j \in S$  通过变异，状态  $i$  转移到状态  $j$  的概率为  $m_{ij} = p_m^{H_{ij}} (1 - p_m)^{N-H_{ij}} > 0$  其中  $H_{ij}$  代表用二进制表示的状态  $i$  和状态  $j$  之间的汉明距离，因此，矩阵  $M$  是正定的。

对于所有的  $i \in S$ ，通过选择，状态不改变的概率之下界为

$$S_{ii} = \frac{\prod_{k=1}^n f(\pi_k(i))}{\left[ \sum_{k=1}^n f(\pi_k(i)) \right]^n} > 0 \quad (6-8)$$

因此，矩阵  $S$  是列可容的。

由引理 6-1 得  $P = CMS$  是正定的。由于每个正定矩阵都是基本的，定理得证。证毕。

**推论 6-1** 参数的取值和定理 6-4 一样的遗传算法是一个遍历的马尔可夫链，即与初始分布无关，该链存在一个惟一的极限分布，并且在任意时刻，位于任意状态的概率非零。

证明 由定理 6-2 和定理 6-4 得证。证毕。

我们注意到马尔可夫链的极限性质与初始状态无关，因此，从理论上讲，遗传算法可以任意初始化。

遍历性对遗传算法的收敛性能有很大的影响。为避免混淆，我们先给出一个遗传算法收敛的精确定义，然后再证明基本遗传算法能否收敛。

**定义 6-5** 令随机变量序列  $Z_t = \max\{f(\pi_k^{(t)}(i)) \mid k = 1, 2, \dots, n\}$  代表在第  $t$  步位于状态  $i$  的种群中个体的最佳适应度，那么遗传算法收敛到全局最优值，当且仅当

$$\lim p\{Z_t = f^*\} = 1 \quad (6-9)$$

其中  $f^* = \max\{f(b) \mid b \in \mathbf{B}^l\}$  为全局最优值。

**定理 6-5** 参数的取值和定理 6-4 一样的遗传算法不收敛到全局最优值。

证明 令  $i$  为状态空间的一个状态，且  $\max\{f(\pi_k(i)) \mid k = 1, 2, \dots, n\} < f^*$ ， $p_i(t)$  为遗传算法在第  $t$  步位于状态  $i$  的概率。显然， $p\{Z_t \neq f^*\} \geq p_i(t)$  从而得到  $p\{Z_t = f^*\} \leq 1 - p_i(t)$ 。由定理 6-2 可知，遗传算法处于状态  $i$  的概率收敛到  $p_i(\infty) > 0$  因此

$$\lim p\{Z_t = f^*\} \leq 1 - p_i(\infty) < 1 \quad (6-10)$$

定理得证。证毕。

**定理 6-6** 一个遍历的马尔可夫链，对于任意的状态  $i$  和  $j$ ，由状态  $i$  转移到状态  $j$  的时间是有限的。

显然，定理 6-5 告诉我们，基本的遗传算法不能保证收敛到全局最优，这一结果令人失望。然而，我们从定理 6-6 得到启示，只要对简单遗传算法做一点改动，每次记录下当前最优解，并将群体状态最前面增加一维存放当前最优解，则改进的遗传算法可望收敛到全局最优。

将遗传算法的种群进行扩展，添加一个超级个体，该个体不参与遗传操作，状态空间的大小由  $2^n$  扩展到  $2^{(n+1)l}$ 。为了描述方便，将超级个体置于种群的二进制表示的最左端，

并用  $\pi_0(i)$  对状态  $i$  的超级个体进行访问。在转移矩阵中，超个体相同的状态的位置连在一起，而且超个体的适应度越大，该状态的位置就越高。

由于超个体不受遗传算子的影响，扩展后的交叉矩阵  $\mathbf{C}^c$ 、变异矩阵  $\mathbf{M}^c$  和选择矩阵  $\mathbf{S}^c$  可被写成分块对角矩阵：

$$\mathbf{C}^c = \begin{bmatrix} \mathbf{C} & & \\ & \mathbf{C} & \\ & & \ddots \\ & & & \mathbf{C} \end{bmatrix}, \mathbf{M}^c = \begin{bmatrix} \mathbf{M} & & \\ & \mathbf{M} & \\ & & \ddots \\ & & & \mathbf{M} \end{bmatrix}, \mathbf{S}^c = \begin{bmatrix} \mathbf{S} & & \\ & \mathbf{S} & \\ & & \ddots \\ & & & \mathbf{S} \end{bmatrix} \quad (6-11)$$

其中  $\mathbf{C}, \mathbf{M}, \mathbf{S}$  为  $2^l \times 2^l$  的方阵 各有  $2^l$  个 因此

$$\mathbf{C}^c \mathbf{M}^c \mathbf{S}^c = \begin{bmatrix} \mathbf{CMS} & & \\ & \mathbf{CMS} & \\ & & \ddots \\ & & & \mathbf{CMS} \end{bmatrix} \quad (6-12)$$

其中  $\mathbf{CMS} > 0$ 。

升级操作由一个“升级”(upgrade)矩阵  $\mathbf{U}$  表示，其作用是将一个包含优于它的超个体的个体的中间状态升级为超个体等于该较优个体的状态。特别地，令

$$b = \operatorname{argmax}\{f(\pi_k(i)) \mid k = 1, 2, \dots, n\} \in \mathbf{IB}^l \quad (6-13)$$

代表在状态  $i$  除超个体外 种群中的最优个体。如果  $f(\pi_0(i)) < f(b)$  那么  $u_{ij} = 1$  其中状态  $j$  定义为  $(b, \pi_1(i), \pi_2(i), \dots, \pi_n(i)) \in S$  否则  $u_{ij} = 0$ 。这样 每一行中有且只有一个元素，而每一列则不然。因为对于状态  $j \in S$  如果  $f(\pi_0(j)) < \max\{f(\pi_k(j)) \mid k = 1, 2, \dots, n\}$  那么对于任意的状态  $i \in S, u_{ij} = 0$ 。换句话说，一个状态要么被升级 要么保持不变，因此升级矩阵可被写成

$$\mathbf{U} = \begin{bmatrix} \mathbf{U}_{11} & & \\ \mathbf{U}_{21} & \mathbf{U}_{22} & \\ \vdots & \vdots & \ddots \\ \mathbf{U}_{2^l,1} & \mathbf{U}_{2^l,2} & \dots & \mathbf{U}_{2^l,2^l} \end{bmatrix} \quad (6-14)$$

其中 子矩阵  $\mathbf{U}_{aa}$  的大小为  $2^l \times 2^l$ 。为了简单起见，假设待优化的问题只有一个全局最优解 那么 只有  $\mathbf{U}_{11}$  为单位矩阵 而其他所有  $\mathbf{U}_{aa} (a \geq 2)$  为对角线上有零元素的单位矩阵。

令  $\mathbf{P} = \mathbf{CMS}$  则遗传算法的转移矩阵为

$$\mathbf{P}^c = \begin{bmatrix} \mathbf{P} & & \\ & \mathbf{P} & \\ & & \ddots \\ & & & \mathbf{P} \end{bmatrix} \begin{bmatrix} \mathbf{U}_{11} \\ \mathbf{U}_{21} & \mathbf{U}_{22} \\ \vdots & \vdots & \ddots \\ \mathbf{U}_{2^l,1} & \mathbf{U}_{2^l,2} & \dots & \mathbf{U}_{2^l,2^l} \end{bmatrix} = \begin{bmatrix} \mathbf{PU}_{11} \\ \mathbf{PU}_{21} & \mathbf{PU}_{22} \\ \vdots & \vdots & \ddots \\ \mathbf{PU}_{2^l,1} & \mathbf{PU}_{2^l,2} & \dots & \mathbf{PU}_{2^l,2^l} \end{bmatrix} \quad (6-15)$$

其中  $\mathbf{PU}_{11} = \mathbf{P} > 0$ 。由于子矩阵  $\mathbf{PU}_{a1} (a \geq 2)$  可以合并起来用一个长方矩阵  $\mathbf{R} \neq 0$  表示，因此我们可以利用定理 6-3 来证明遗传算法的收敛性。

**定理 6-7** 参数的取值和定理 6-4 一样，并在选择之后保留当前最优值的遗传算法收敛到全局最优值。

证 明 子 矩 阵  $\mathbf{PU}_{11} = \mathbf{P} > 0$  集中了包含全局最优的超个体的状态（全局最优状态）

之间的转移概率。由于  $P$  是基本随机矩阵且  $R \neq 0$  定理 6-3 保证了遗传算法收敛到所有非全局最优状态的概率为零。从而收敛到全局最优状态的概率为 1 即

$$\lim p\{Z_i = f^*\} = 1 \quad (6-16)$$

算法收敛到全局最优值。证毕。

定理 6-7 证明了在选择之后保留最优值的遗传算法可以收敛到全局最优值。用类似的方法，还可证明在选择之前保留最优值的遗传算法也可收敛到全局最优值。

定理 6-8 参数的取值和定理 6-4 一样，并在选择之前保留当前最优值的遗传算法收敛到全局最优值。

证明 转移矩阵为  $P^C = T^C U S^C$  其中  $T^C = C^C M^C$ 。令  $T = CM$  则

$$P^C = \begin{bmatrix} T & & \\ & T & \\ & & \ddots \\ & & & T \end{bmatrix} \begin{bmatrix} U_{11} & & \\ U_{21} & U_{22} & \\ \vdots & \vdots & \ddots \\ U_{2^t,1} & U_{2^t,2} & \cdots & U_{2^t,2^t} \end{bmatrix} \begin{bmatrix} S & & \\ & S & \\ & & \ddots \\ & & & S \end{bmatrix} = \begin{bmatrix} TU_{11}S & & \\ TU_{21}S & TU_{22}S & \\ \vdots & \vdots & \ddots \\ TU_{2^t,1}S & TU_{2^t,2}S & \cdots & TU_{2^t,2^t}S \end{bmatrix} \quad (6-17)$$

其中  $TU_{11}S = TS = P > 0$ 。同样地，子矩阵  $TU_{a1}S (a \geq 2)$  可以合并起来用一个长方矩阵  $R \neq 0$  表示。仿照定理 6-7，就可以证明遗传算法收敛到全局最优值。证毕。

## 6.5 遗传算法面临的问题及改进算法

遗传算法来源于生物进化论和群体遗传学，缺乏严格的数学基础，收敛性证明比较困难，虽然在上一节中利用马尔可夫链的性质证明了保留最优值的遗传算法最终能收敛到全局最优值，但收敛到最优值所需的时间可能很长，而且许多改进算法的证明仍有困难。目前，关于遗传算法最终收敛到全局最优值的时间复杂度仍是有待解决的问题，需要寻求更有效的分析手段和严格的数学证明。另外，遗传算法中，关于如何确定种群大小、选择方式、编码方式和遗传算子的概率等仍然需要进一步研究，而这些对遗传算法的性能和结果的质量至关重要。

遗传算法中另一个不可忽视的现象是未成熟收敛问题，主要表现在：群体中所有的个体都陷于同一极值而停止进化；接近最优解的个体总是被淘汰，进化过程不收敛。

导致未成熟收敛的主要因素如下：在进化初始阶段生成具有较高适应度的个体  $X$  在基于适应度比例的选择下，淘汰了其他个体，留下的大部分个体与  $X$  一致；相同的两个体交叉，未能生成新个体；经变异生成的新个体适应度虽高但可能数量少，被淘汰的概率很大。

另外，遗传算法虽然具有较好的全局收敛性，但局部优化主要靠变异来完成，收敛性较差，当找到次优解后，很难迅速收敛到最优解。

为了更广泛地应用遗传算法解决实际问题，必须从各方面改善遗传算法的性能。人们提出了各种各样的改进算法，大致可从以下几方面着手改进：

### 1. 简单保留最优值的遗传算法

上一节已经证明，种群为有限样本集时，经典遗传算法不依概率 1 收敛到最优解，而

保留最佳个体的经典遗传算法可依概率 1 收敛到最优解。

## 2. 选择算子的改进

目前常用的几种选择方法有：

### (1) 适应度比例方法

适应度比例方法是遗传算法中最基本也是最常用的选择方法，又叫轮盘赌选择方法。在表 6-1 中，我们采用的就是这种方法（如图 6-2 所示），首先计算种群中所有个体的适应度总和  $\sum f$ ，再计算每个个体适应度所占的比例  $f_i / \sum f$ ，并以此作为每个个体的选择概率，然后将轮盘按各个个体的选择概率进行分配，转动轮盘  $N$  次，就可以得到选择结果。

### (2) 期望值方法

由表 6-1 可以看出，在轮盘赌选择方法中，当个体数不太多的时候，依据产生的随机数进行选择不一定正确反映个体适应度，适应度高的个体可能被淘汰，而适应度低的个体有可能被选择。为解决这一矛盾，人们采用期望值方法。

采用这种方法时，首先计算每个个体在下一代生存的期望数目  $M_i$ 。

$$M_i = \frac{f_i}{\bar{f}} = \frac{f_i}{\sum f_i / N} \quad (6-18)$$

若某个体被选中，并要参与配对交叉，则将生存期望数减去 0.5 若不参与配对交叉 则将该个体的生存期望数减去 1。在上述两种情况中，若个体的生存期望数小于零，则该个体不参与选择。

### (3) 排序选择方法

根据适应度大小对种群中的个体进行排序，然后把事先设计好的概率表按序分配给每个个体，因而选择概率和适应度无直接关系而仅与序号有关。采用这种方法，需要事先确定选择概率和序号的关系。

### (4) 联赛选择方法

从种群中任意选择一定数目的个体（称为联赛规模），其中适应度最高的个体保存到下一代。这一过程反复执行，直到下一代的个数达到预先设定的数目为止。联赛规模一般为 2。

### (5) 排挤方法

在采用覆盖种群模式的情况下，排挤方法可描述为：设定参数 CF 从种群中随机地挑选 CF 个个体组成个体集（新的个体不包括在内）；从这个个体集中淘汰一个个体，该个体与新个体的汉明距最小。这种方法的好处是可以提高种群的多样性。

## 3. 交叉算子的改进

目前常用的几种交叉方法有：

### (1) 一点交叉

在个体串中随机设定一个交叉点，该点前或后的两个个体的部分结构进行互换，生成两个新个体。表 6-1 中采用的就是一点交叉的方法。

### (2) 两点交叉

两点交叉与一点交叉类似，只是需要在个体串中随机设定两个交叉点。一个两点交叉的例子如下：

个体 A	01		100		100	————→	01001100
个体 B	10		001		010	————→	10100010
			交叉点 1		交叉点 2		

### (3) 多点交叉

多点交叉是前两种交叉的推广，在两个串中设立多个交叉点。

### (4) 均匀交叉

均匀交叉是通过设立屏蔽字，来决定新个体每一位上的基因继承哪个旧个体的基因。

一个均匀交叉的例子如下：

旧个体 A	00101101
旧个体 B	11101100
屏蔽字	01000101
新个体 A'	01101100
新个体 B'	10101101

可见 当屏蔽字为 0 时 新个体 A' 的基因来自旧个体 A 而当屏蔽字为 1 时 新个体 A' 的基因来自旧个体 B。

一般讲，所选交换点越多，原来个体的变化就会越大，产生新个体和丢失旧个体的机会也越大。

除了采用不同的交叉方式外，对如何选择交叉伙伴也是一个很重要的问题，选择方式不同，那么在下一代产生的个体成员的质量就不相同。通过研究，人们提出了近亲繁殖、远亲繁殖和类繁殖等设想。

## 4. 变异算子的改进

变异概率  $p_m$  的大小对遗传算法的性能有较大影响。变异概率取在一定范围内，才能使遗传算法有效搜索解空间。当  $p_m$  太小时，每代发生的变异太少，这些变异个体极易通过选择算子而丢失 造成遗传算法的过早收敛 不易达到最优 当  $p_m$  太大时，每代发生的变异太多，这抵消了选择算子的作用，造成遗传算法不能在已发现的较优区域有效搜索。

为了改进遗传算法的适应能力，人们将变异概率由固定改为依环境而变。当种群差异大时，通过交叉操作即可搜索到整个解空间，此时变异率可以较小；种群差异小时，通过交叉产生的新模式有限，需要通过增大变异概率改善遗传算法搜索能力。例如，变异概率依两个父本的汉明距的大小而变，当汉明距较小时，变异概率较大，反之则较小。另外，变异概率还可随种群熵、位熵而变。这种自适应的变异概率，可以在一定程度上防止未成熟收敛的发生。

## 5. 小生境方法

在自然界中，生物总是倾向于与自己特征类似的生物群生活在一起，并与同类交配繁衍后代，这种现象对于进化过程的形成具有积极作用。在生物学中，将特定环境下物种的组织功能称为小生境（niche）。针对单目标多解函数的优化问题，借用生物中的小生境概念可使遗传算法得到改进。

基本的遗传算法在一次搜索过程中只能得到目标函数的一个极值点。这时，个体间的竞争不受限制。进化过程中分散在较小极值点附近的个体都可能被淘汰。此外，由于群体

规模有限，在算法的选择过程中不可避免地出现随机错误。因此，基本遗传算法不能在一次搜索过程中找到目标函数的多个极值点。

如果能在群体中长时间保持多个物种，就有可能在一次搜索过程中得到目标函数的多个极值点。为此，改进的遗传算法必须具有维护群体内小规模低适应度物种生存，并具有防止随机错误出现和纠正随机错误的能力。所谓小生境遗传算法实现上述要求的途径主要有两条：一是动态调整物种的适应度，利用物种间的竞争保持群体内各物种之规模大体稳定；另一是限制种群中可进行竞争的个体对之间的相似性，从而维持各物种规模之稳定。

根据算法保护低适应度物种采取的不同方法，小生境遗传算法大体上可分为两类：基于适应度调整的方法和基于直接竞争限制的方法。下面简要介绍这两类方法中具有代表性的算法调整策略基本思想，即共享机制调整策略和排挤机制调整策略。

共享机制调整策略是将一个大的解空间分成若干个解的子空间，然后在每个子空间进行求解，适用于解是均匀分布的多解函数，或者讲周期函数。在这种方法中，通过定义共享函数来确定每个个体在种群中的共享度。一个个体的共享度等于该个体与种群中其他个体的共享函数值的总和。共享函数是两个个体之间关系密切程度（基因型的相似性或表现型的相似性）的函数，当个体间关系比较密切时，共享函数值较大，反之共享函数值较小。定义两个个体  $x_i$  与  $x_j$  之间的距离的度量为  $d_{ij} = d(x_i, x_j)$  共享函数  $Sh(d)$  具有如下性质：

$$(1) 0 \leq Sh(d) \leq 1;$$

$$(2) Sh(0) = 1;$$

$$(3) \lim_{d \rightarrow \infty} Sh(d) = 0.$$

一种可能的函数形式为

$$Sh(d) = \begin{cases} 1 - \left[ \frac{d}{\sigma_{sh}} \right]^a, & d < \sigma_{sh} \\ 0, & d \geq \sigma_{sh} \end{cases} \quad (6-19)$$

其中  $\sigma_{sh}$  是一个常数因子，被称为共享因子。个体的共享适应度为

$$f'_i = \frac{f_i}{m'_i} \quad (6-20)$$

其中  $f_i$  为个体  $x_i$  原来的适应度， $m'_i$  为个体  $x_i$  的共享度

$$m'_i = \sum_{j=1}^N Sh(d_{ij}) = \sum_{j=1}^N Sh[d(x_i, x_j)] \quad (6-21)$$

可以看出，基本的遗传算法选择算子只能使个体向高适应度的可行域聚集，不能限制种群内某一类“物种”的无控制增长；而这里给出的共享机制算法则可依据群体在可行域上的分布密度调整个体适应度，将聚集在一起的个体共享适应度适当调低，以便通过选择使过于集中的个体能够分散开，从而在目标函数多个极值点的周围保持物种稳定。

排挤机制调整策略是一种基于直接竞争限制的思想方法。这种方法将竞争和替换限制在同物种个体之间，延长低适应度物种的生存时间，减小发生随机错误的可能，从而可以在算法的一次搜索过程中得到目标函数的多个极值点。基于排挤机制的小生境算法与基本的遗传算法之主要区别表现在：前者在算法的一个循环周期中只产生几个（或一个）

新个体，并用它们替换老群体中的部分个体；而后者在算法的一个循环周期中将整个老群体用它们的子女替换掉，这种算法可称为整代进化的遗传算法，而排挤策略的小生境算法可称为“稳态的遗传算法”。限于篇幅，此处不再讨论这种算法的具体步骤，可参看有关文献<sup>[11]</sup>。

目前，小生境遗传算法已经日益受到人们的重视。例如，在雷达、声纳、地震信号检测、射电天文和医学成像等学科领域广泛采用阵列信号处理技术，其中，某些处理方法希望在一次搜索过程中找到目标函数的多个极值点。著名的 MUSIC 阵列信号处理算法就是要求寻找 MUSIC 谱中多个谱峰的位置，对于此类问题，小生境遗传算法将显示其突出优点。

#### 6. 防止未成熟收敛的一些技巧

主要有以下一些方法，可防止未成熟的收敛。

- (1) 提高变异概率，在进化的初始阶段，可以加强遗传算法随机搜索能力。
- (2) 调整选择概率，把选择概率本身也作为个体来进行优化。
- (3) 对适应度进行调整。
- (4) 维持群体中个体的多样性：增加群体规模；实施局部化；实施单一化；增大交叉个体距离。

#### 7. 遗传算法与其他优化方法相结合

为了弥补遗传算法的不足之处，人们尝试将遗传算法与其他优化算法相结合，并取得了较好的结果。例如，在遗传算法框架中，适度地引入其他的局部搜索方法用以改善遗传算法的局部搜索能力在理论和实际上都是十分必要的。其中，将遗传算法和模拟退火算法相结合，遗传算法的主要任务是为模拟退火算法筛选出一个好的算法构形或称变量组态。通俗地讲，通过遗传算法来选择解空间中的优良点，以便让模拟退火算法从一些优良点开始搜索。这对于提高算法的速度和改善所得问题解的质量都能产生较好的效果。

#### 8. 引入并行处理技术

在介绍遗传算法的特点时已经指出，遗传算法具有并行性，个体成员参数的计算及其对应的目标函数值的计算、个体交叉、变异等运算都可以并行进行，所以为引入并行算法技术提供了条件。引入并行技术，可以大大提高遗传算法的速度，这对于解决实时问题具有重大意义。目前，主要的并行模型有粗粒度孤岛模型和细粒度邻域模型两种。其中，粗粒度孤岛模型的算法流程步骤如下。

- (1) 随机产生一个初始群体并将它分成  $N$  个子群体；
- (2) 并发地对每个子群体执行下列步骤 (3) 和 (4)；
- (3) 在所给定的进化代内对子群体执行基因操作，包括选择、交叉和变异；
- (4) 邻域间的个体迁移，形成新一代子群体；
- (5) 若不满足结束条件，则转去 (2)。

而细粒度邻域模型的构成原理是借助分布式计算的通用框架实现遗传算法的基本操作，也即用“细胞自动机”与遗传算法相结合完成细粒度邻域模型的并行运算。此处不再详述。

## 9. 优劣复取舍遗传算法

在文献[12]、[14]中，作者提出了“优劣复取舍遗传算法”(genetic algorithm based on better-worse re-selection, BWRGA)。这种算法以增大种群适应度为目标，经理论推证导出了遗传算子作用于个体产生结果的取舍依据——即优劣复取舍原则，有选择地保留操作结果，改进了遗传算法的收敛性能。下面介绍文献[14]的推证。

设种群  $G = \{x_i \mid i = 1, 2, \dots, N\}$  其中  $x_i \in \{0, 2^{-l}, \dots, 1 - 2^{-l}\}$ 。定义种群适应度为

$$Q(G) = \sum f(x_i) \quad (6-22)$$

种群大小为  $N$  经过选择 种群由  $G$  变为  $G'$ 。若选择概率为

$$p(x'_i = x_i) = \frac{f(x_i)}{\sum_j f(x_j)} \quad (6-23)$$

则种群  $G'$  适应度的期望值为

$$E[Q(G')] = \sum_i \frac{Nf^2(x_i)}{\sum_j f(x_j)} \quad (6-24)$$

于是

$$E[Q(G')] - Q(G) = \sum_i \frac{Nf^2(x_i)}{\sum_j f(x_j)} - \sum_i f(x_i) = \frac{N \sum_i f^2(x_i) - \left[ \sum_i f(x_i) \right]^2}{\sum_j f(x_j)} \geq 0 \quad (6-25)$$

可见，从概率意义上说，选择的作用是使种群整体适应度增大。

取  $x_i$  进行遗传算子操作，结果为  $x'_i$ 。 $G$  中  $x_i$  变为  $x'_i$  经过选择得到新种群  $G'$ 。考察  $G$  与  $G'$  两个种群的适应度的关系。由于用  $x'_i$  替代  $x_i$ ，上式中左端第一项修改为

$$E[Q(G')] = \frac{N \left[ \sum_j f^2(x_j) + f^2(x'_i) - f^2(x_i) \right]}{\sum_j f(x_j) + f(x'_i) - f(x_i)} \quad (6-26)$$

于是

$$\begin{aligned} E[Q(G')] - Q(G) &= \frac{N \left[ \sum_j f^2(x_j) + f^2(x'_i) - f^2(x_i) \right]}{\sum_j f(x_j) + f(x'_i) - f(x_i)} - \sum_j f(x_j) \\ &= \frac{N \left[ \sum_j f^2(x_j) + f^2(x'_i) - f^2(x_i) \right] - \left\{ \left[ \sum_j f(x_j) \right]^2 + \left[ \sum_j f(x_j) \right] [f(x'_i) - f(x_i)] \right\}}{\sum_j f(x_j) + f(x'_i) - f(x_i)} \\ &= \frac{N \left[ \sum_j f^2(x_j) \right] - \left[ \sum_j f(x_j) \right]^2 + N[f^2(x'_i) - f^2(x_i)] - \left[ \sum_j f(x_j) \right] [f(x'_i) - f(x_i)]}{\sum_j f(x_j) + f(x'_i) - f(x_i)} \\ &= \frac{\left\{ N \left[ \sum_j f^2(x_j) \right] - \left[ \sum_j f(x_j) \right]^2 \right\} + N[f(x'_i) - f(x_i)][f(x'_i) + f(x_i) - \bar{f}(G)]}{\sum_j f(x_j) + f(x'_i) - f(x_i)} \end{aligned}$$



$$= \frac{N \sum_j f^2(x_j) - \left[ \sum_j f(x_j) \right]^2}{\sum_j f(x_j) + f(x'_i) - f(x_i)} + \frac{N[f(x'_i) - f(x_i)][f(x'_i) + f(x_i) - \bar{f}(G)]}{\sum_j f(x_j) + f(x'_i) - f(x_i)} \quad (6-27)$$

其中  $\bar{f}(G) = \frac{\sum_j f(x_j)}{N}$  为种群的平均适应度。这样上式可改写为

$$E[Q(G')] - Q(G) = \frac{N \sum_j f^2(x_j) - \left[ \sum_j f(x_j) \right]^2}{\sum_j f(x_j) + f(x'_i) - f(x_i)} + \frac{N[f(x'_i) - f(x_i)][f(x'_i) + f(x_i) - \bar{f}(G)]}{\sum_j f(x_j) + f(x'_i) - f(x_i)} \quad (6-28)$$

等式右端的第一项总是大于等于 0，大体相当于单纯进行选择使种群适应度产生的改进。由于进化的目标是使种群适应度的改善尽可能大，因此，必须考察右端第二项的符号，如果符号为正，保留遗传算子的操作结果，反之，则不保留操作结果。

对于多个串的操作结果，经简单修改得到

$$E[Q(G')] - Q(G) = \frac{N \sum_j f^2(x_j) - \left[ \sum_j f(x_j) \right]^2}{\sum_j f(x_j) + \sum_k [f(x'_{i_k}) - f(x_{i_k})]} + \frac{\sum_k N[f(x'_{i_k}) - f(x_{i_k})][f(x'_{i_k}) + f(x_{i_k}) - \bar{f}(G)]}{\sum_j f(x_j) + \sum_k [f(x'_{i_k}) - f(x_{i_k})]} \quad (6-29)$$

根据上面的推导，得到如下的以增大种群适应度为目标的替换规则，即优劣复取舍原则。

(1) 产生好结果  $f(x'_i) > f(x_i)$  若产生结果的父本的适应度  $f(x_i) \geq f(G)/2$  (即父本属于较好样本)，结果予以保留。

(2) 产生好结果  $f(x'_i) > f(x_i)$  若结果适应度  $f(x'_i) < f(G)/2$  结果不保留。

(3) 产生差结果  $f(x'_i) < f(x_i)$ ：若产生结果的父本的适应度  $f(x_i) \leq f(G)/2$  (即父本属于较差样本)，结果予以保留。

(4) 产生差结果  $f(x'_i) < f(x_i)$  若结果适应度  $f(x'_i) > f(G)/2$  结果不保留。

在实施上述取舍原则时，又可分为严格保留与广义保留两种操作方法，分别按以下规则执行。

(5) 严格保留的优劣复取舍原则：仅满足保留条件的保留，其他情况不保留。这种情况下  $E[Q(G')] > Q(G)$  为严格不等式。

(6) 广义保留的优劣复取舍原则：满足保留条件的保留，满足不保留条件的不保留，其他情况随机处理。

从以上分析可以看出，由于全面考虑了式（6-29）中各项的作用，利用优劣复取舍原则可使种群适应度获得尽可能大的改善。

优劣复取舍原则突破了人们长期以来对遗传算法的一个直觉思想，即通过遗传算子所产生的个体并不是单纯的适应度越高，其生存的可能性就越大，而是还与产生该个体父本的适应度以及种群的平均适应度有关。即使子本的适应度好于父本的适应度，但如果子本的适应度与种群的平均适应度相比较差的话，该子本仍然不能替代父本；而如果父本的适应度与种群的平均适应度相比较差的话，即使子本的适应度低于父本的适应度，子本仍然可以替代父本。

在文献[13]中，作者还给出了优劣复取舍遗传算法的收敛定理，对一些典型问题进行了实例计算，并和简单的遗传算法相对照，比较了两种算法在收敛过程中种群平均适应度及种群个体分布情况，验证了加入优劣复取舍原则的遗传算法的效果。

优劣复取舍遗传算法的不足之处是仅以提高种群适应度为优化目标，忽视了遗传算子保持种群多样性的重要性，易出现未成熟收敛现象，需要与其他改进方法相互结合，具体的改进算法可参看文献[13]。

## 6.6 遗传算法与人工神经网络的结合

人工神经网络与遗传算法都是将生物学原理应用于计算智能研究的仿生学理论成果。遗传算法是从自然界生物进化的机制获得启示，而人工神经网络则是对人脑或动物神经网络若干基本特征的抽象和模拟。因此，它们在信息处理的方式和时间上存在着较大差异。通常，神经系统的变化比较快，而生物的进化却需以世代的尺度衡量。GA与ANN各自有其特点和长处。近年来，越来越多的学者尝试将GA与ANN结合，希望通过二者的有机结合，充分利用二者的长处，寻找更为有效的解决问题方法。一般来说，GA与ANN的结合可以在三个层面上进行，这就是神经网络连接权的进化、神经网络结构的进化和神经网络学习规则的进化。下面分别介绍它们的基本原理<sup>[15]</sup>。

### 6.6.1 神经网络连接权的进化

这种结合方式的基本原理是固定网络结构，利用GA训练网络权重。由于前向网络用BP方法训练存在可能陷入局部极小等不足，可以用GA代替BP作为前向网络的学习算法。以GA作为学习算法主要应解决编码方案问题，即网络权重和染色体之间的相互映射问题。

借助遗传算法优化神经网络连接权的算法步骤如下。

（1）选定网络结构和学习规则。随机产生一组权重值，利用某种编码方案对每个权重值进行编码。将网络中的权重值依次排列构成码链。每个码链代表网络的一种权重分布状态，一组码链则代表一组不同权重值的神经网络。

（2）计算在每个对应码链下神经网络的误差函数，从而给出遗传算法所需的适应度函数，误差愈小适应度值愈高。

（3）选择若干适应度函数值最大的个体构成父本。

(4) 利用交叉、变异等遗传操作算子对当前一代群体进行处理，产生出新一代群体。

(5) 重复上述 (2)、(3)、(4) 步骤，使权重值分布不断进化，直至达到训练目标为止。

将基于遗传算法的神经网络连接权进化计算与基于梯度下降的 BP 算法相比较，有如下特点。

(1) 遗传进化方法可以实现全局搜索，不需误差函数的梯度信息，不必考虑误差函数是否可微，这是它的突出优点。如果在训练过程中容易获取梯度信息，BP 算法则有可能在速度上优于遗传进化方法。

(2) 两种算法的结果都对计算过程中用到的算法参数很敏感。BP 算法的结果还对网络的初始状态有密切的依从关系。

(3) 遗传进化方法擅长全局搜索，而 BP 算法在局部搜索时显得比较有效。

(4) 受编码精度的限制，遗传进化方法有时很难得到非常高的训练精度。

综上所述，若将两种算法相结合构成所谓混合训练算法，有可能相互取长补短获得较好的训练结果。例如，可借助遗传算法善于发现较好空间区域之特点，先进行计算，给出初值，然后再利用 BP 算法对权重进行精调，搜索最优解。一般讲，混合训练的效率和效果比单独用遗传进化或 BP 训练方法要有明显改善。

神经网络连接权进化的计算实例将在稍后给出。

## 6.6.2 神经网络结构的进化

神经网络结构包括网络的拓扑结构（连接方式）和节点转移函数两方面的内容。人们总是期望以简单的网络结构实现所需的信号处理功能，并尽可能达到较高的性能指标。然而，神经网络结构的选择与设计还未能找到有效、合理的方法。目前，大多依赖于设计者的主观经验，用试探、比较的办法适当选择优化的网络结构，缺乏系统、严格的理论指导。利用遗传算法设计神经网络可根据某些性能评价准则，如学习速度、泛化能力或结构复杂程度等搜索结构空间中满足问题要求的最佳结构。利用遗传算法设计神经网络的关键问题之一仍然是如何选取编码方案。

借助遗传算法优化神经网络结构的算法步骤如下。

(1) 随机产生若干个不同结构的神经网络，对每个结构编码，每个码链对应一个网络结构， $N$  个码链构成种群。

(2) 利用多种不同的初始连接权值分别对每个网络进行训练。

(3) 计算在每个对应码链下神经网络的误差函数，利用误差函数或其他策略（如网络的泛化能力或结构复杂程度）确定每个个体的适应度函数。

(4) 选择若干适应度函数值最大的个体构成父本。

(5) 利用交叉、变异等遗传操作算子对当前一代群体进行处理，产生出新一代群体。

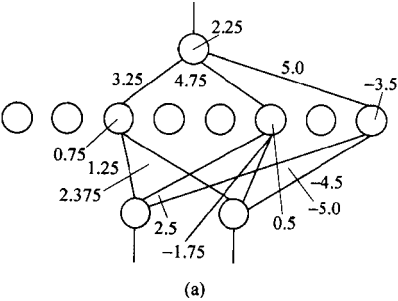
(6) 重复上述 (2)~(5) 步骤，直到群体中的某个个体对应一个网络结构能满足要求为止。

**例 6-2** 利用遗传进化方法设计一个前向多层神经网络。给定条件如下：输入信号为二维（输入层取两个神经元），限定隐层神经元最大可能值为 8 个，输出层神经元只有一个。自动完成结构与连接权之设计。

解 图 6-3(a) 示意给出网络结构图形。首先考虑编码方案。基因串包括以下五个部分：索引串、隐层神经元输入权重编码串、隐层神经元阈值编码串、隐层神经元输出权重编码串和输出层神经元阈值编码串。

在本例中，权重和阈值采用 8 位编码。其中第 1 位为符号位。第 2 位到第 8 位表示权重和阈值的大小，其取值范围为  $[0, 8)$ ，因而取值间隔为  $1/16$ 。这样，对一个二维输入，限定隐层神经元个数最大为 8 个，输出为一个神经元的前向网络，经编码后，得到的基因串长度为 272 位。

按照图 6-3(a) 标注的十进制参数示例可以给出对应的编码后基因串，见图 6-3(b)。



索引串: 00100101  
 隐层神经元输入权重: \*\*\*\*\* 10010100 10100110  
 \*\*\*\*\* 10101000 01100100 \*\*\*\*\* 00110000  
 00111000  
 隐层神经元阈值: \*\*\*\*\* 10001100 \*\*\*\*\* 10001000 \*\*\*\*\*  
 01001000  
 隐层神经元输出权重: \*\*\*\*\* 10110100 \*\*\*\*\* 11001100  
 \*\*\*\*\* 11010000  
 输出神经元阈值: 10100100  
 (整个基因串为各部分依次连接)

(b)

图 6-3 例 6-2 的网络结构和基因串

在确定编码方式之后，另一个重要问题就是确定适应度函数，用于评价基因串的适应度，从而评价与基因串对应的神经网络的好坏。在本例中，综合考虑训练误差和隐层神经元个数（从而考虑网络的泛化能力）。前向网络应保证一定训练误差的条件下，以尽可能少的神经元实现。我们采用了如下的适应度函数：

$$fitness(C) = f_{\epsilon}(\epsilon) / f_n(n)$$

$$f_{\epsilon}(\epsilon) = \begin{cases} \frac{1}{\epsilon_{min}}, & \epsilon < \epsilon_{min} \\ \frac{1}{\epsilon}, & \epsilon > \epsilon_{min} \end{cases}$$

$$f_n(n) = \begin{cases} 1, & n < n_T \\ n, & n \geq n_T \end{cases}$$

$f_{\epsilon}(\epsilon)$  反映总和训练误差的影响，且当总和训练误差达到精度要求的情况下， $f_{\epsilon}(\epsilon)$  饱和。 $f_n(n)$  约束隐层神经元个数，保证以尽可能少的隐层单元实现网络， $n_T$  为可接受的神

经元个数下限。这样,  $fitness(C)$  综合了训练精度要求和网络泛化能力的要求。

在交叉算子运行规律的选择方面, 我们注意到由于编码串有实际的物理意义, 串中不同段对应于神经网络的不同参数, 所以采用逐段单点交叉的方式, 即对每一个串中有物理意义的段独立进行单点交叉。

### 6.6.3 神经网络学习规则的进化

在前面讨论的神经网络训练问题中, 学习规则都是事先设定的, 未必合理。可以利用 GA 来设计 ANN 的学习规则, 通过进化使之适应环境的要求, 也可发现新的规则。一般讲, 学习规则的进化包括学习参数的进化和学习规则的进化两个方面。为实现这种进化, 关键问题也是要解决好如何将学习规则进行编码。

借助遗传算法优化神经网络学习规则的算法步骤如下。

(1) 产生若干个体, 每个个体对应一个学习规则, 利用某种编码方案对每个学习规则进行编码。

(2) 构作一个训练集, 其中, 每个元素对应一个神经网络, 选定 (或随机产生而确定) 每个神经网络的结构和初始连接权, 然后对训练集中的元素分别用每个学习规则进行训练。

(3) 计算每个学习规则的适应度。

(4) 选择若干适应度函数值最大的个体构成父本。

(5) 利用交叉、变异等遗传操作算子对当前一代群体进行处理, 产生出新一代群体。

(6) 重复上述 (2) ~ (5) 步骤, 直到群体中的某个个体 (对应一个学习规则) 能满足要求为止。

在学习规则进化方式中, 一种简单的情况是学习参数的进化, 这时, 基本的学习规则 (如连接权调整步骤) 已选定, 只要求优选此训练规则中的参数值。这些参数在调整网络行为中往往产生重要作用。当学习规则本身需要适应网络环境而进行优化时, 对学习规则编码方案的选取将比较复杂。

目前, 对神经网络学习规则进化的研究尚不成熟, 这是一个具有发展潜力的研究领域, 日益受到人们的重视。

以上分别讨论了利用遗传算法优化神经网络连接权、结构和学习规则三个层面上的相结合原理。除此之外, 遗传算法与人工神经网络的结合还有其他各种形式。例如, 利用遗传算法进行特征提取, 优化神经网络的输入信号, 此时, 遗传算法完成了神经网络输入数据的预处理功能。特征提取的成功与否对模式分类的结果起关键作用。利用 GA 可以选择数据集的参数和特征的尺度因子, 缩小数据间类内差别, 扩大类间差别, 改善 ANN 分类性能。

此外, 还可将神经网络作为遗传算法运行中的适应度评价器 (或称神经网络性能预测器), 可自动计算适应度值。下面给出这种结合方法的应用实例。

**例 6-3 利用遗传-神经网络算法优化 ATM (异步转移模式) 通信网链路容量控制<sup>[50]</sup>**

在本例中, 我们将利用遗传算法优化 ATM 网络链路容量之分配, 而神经网络作为性

能预测器嵌入遗传算法运行过程之中。

在实际的优化问题中，往往很难给出优化的目标函数，只能知道目标函数值的大小与哪几个参数有关。这个问题可以利用神经网络的自学习功能解决。例如，在 ATM 链路容量的用户分配中，一般选取网络性能作为分配的优化目标，而用户业务量和网络性能的未知关系可用人工神经网络解决。当给定观察的业务量和相应网络性能作为学习样本，神经网络可自动学习，掌握其中的非线性关系，而不需对网络进行复杂的分析，当用户业务参数和网络情况发生改变时，ANN 还能自适应地控制，同时 ANN 大规模并行处理的方式将大大提高处理速度，满足 ATM 控制的要求。因此，为获得较好的优化结果，当给定一组链路容量分配值和此时的业务量情况，利用 ANN 预测器估计此时相应的网络服务质量，如呼损率、时延、链路利用率等，然后利用遗传算法进行优化分配，直到满足优化目标。

采用链路利用率为优化目标，即

$$\max \{u_1, u_2, \dots, u_n\} - F\{u_1, u_2, \dots, u_n\} \rightarrow \min \quad (6-30)$$

式中  $u_i$  是第  $i$  条逻辑链路 ( $i = 1, 2, \dots, n$ ) 的利用率，它是链路容量  $V_i$  和呼叫到达率  $a_i$  的未知非线性函数，

$$u_i = f(V_i, a_i) \quad (6-31)$$

$n$  是逻辑链路的总数； $F\{u_1, u_2, \dots, u_n\} = \min \{u_{1'}, u_{2'}, \dots, u_{n'}\}$  其中第  $j$  条 ( $j = 1', 2', \dots, n$ ) 逻辑链路是和利用率最大的逻辑链路共用某一条物理链路的逻辑链路。链路容量的分配受限于

$$\sum_j t_{ij} V_i = C_j \quad (6-32)$$

式中  $C_j$  ( $j = 1, 2, \dots, M$ ) 表示物理链路的容量； $M$  表示干线数； $t_{ij}$  定义为

$$t_{ij} = \begin{cases} 1, & \text{当逻辑链路 } i \text{ 通过物理链路 } j \\ 0, & \text{其他} \end{cases}$$

为简化目标函数，在容量分配中保证式 (6-32) 得到满足，则不受限的目标函数为

$$E(V) = \max \{u_1, u_2, \dots, u_n\} - F\{u_1, u_2, \dots, u_n\} \quad (6-33)$$

为达到优化目标，需使目标函数 (6-33) 达到最小。

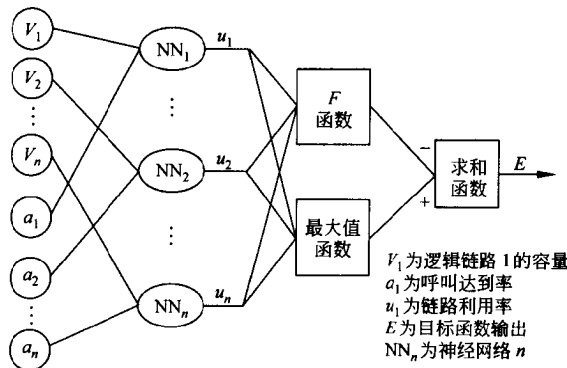


图 6-4 神经网络预测器结构图

根据式(6-33),构造神经网络预测器如图6-4。图中所示的结构由3层前向ANN预测器和一些特殊的简单神经元构成。这些特殊神经元的输出函数可以是最大值函数、 $F$ 函数和简单的线性函数等。由神经网络预测器预测每条逻辑链路的利用率 $u_i$ ,再通过遗传优化算法进行链路的优化分配。图6-5给出了神经网络嵌入遗传算法的框图。

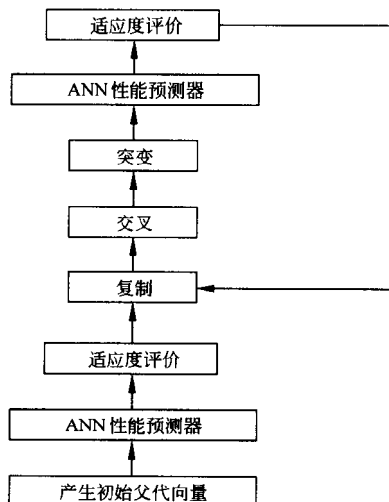


图 6-5 嵌入 ANN 性能预测器的遗传算法框图

## 参 考 文 献

- [1] Holland John H. Adaptation in natural and artificial systems; an introductory analysis with applications to biology, control, and artificial intelligence. Cambridge;The MIT Press, 1992
- [2] Angeline Peter J. Saunders Gregory M, and Pollack Jordan B. An evolutionary algorithm that constructs recurrent neural networks. IEEE Transactions on Neural Networks,1994,5(1):54 ~ 65
- [3] Maniezzo Vittorio. Genetic evolution of the topology and weight distribution of neural networks. IEEE Transactions on Neural Networks, 1994 . 5(1): 39 ~ 53
- [4] Rudolph Gunter. Convergence analysis of canonical genetic algorithms. IEEE Transactions on Neural Networks, 1994. 5(1): 96 ~ 101
- [5] Qi Xiaofeng, Palmieri Francesco. Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space part I: basic properties of selection and mutation. IEEE Transactions on Neural Networks, 1994,5(1) :102 ~ 119
- [6] Qi Xiaofeng, Palmieri Francesco. Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space part II: analysis of the diversification role of crossover. IEEE Transactions on Neural Networks, 1994,5(1) :120 ~ 129
- [7] Goldberg D E. Genetic algorithms in Search Optimization and Machine Learning. MA: Addison-Wesley, 1989
- [8] 孟庆春. 基因算法及其应用. 山东: 山东大学出版社, 1995
- [9] 陈国良, 王煦法, 庄镇泉等. 遗传算法及其应用. 北京: 人民邮电出版社, 1996
- [10] 恽为民, 席裕庚. 遗传算法的运行机理分析. 控制理论与应用, 1996, 13(3):297 ~ 304
- [11] 邹燕明. 小生境遗传算法的研究与应用: [博士学位论文]. 北京: 北京理工大学, 1999. 4
- [12] 张宇. 利用计算智能技术的 ATM 网络流量控制研究: [博士学位论文]. 北京: 清华大学, 1998. 8
- [13] 陈文霞. 遗传与神经网络优化算法及其在 ATM 网络控制中的应用: [博士学位论文]. 北京: 清华大学, 2001
- [14] 陈文霞, 郑君里, 张宇. 优劣复取舍遗传算法. 清华大学学报, 2000, 40(7):77 ~ 80
- [15] Yao Xin. Evolving artificial neural networks. Proceedings of the IEEE 87(9) . September 1999 : 1423 ~ 1447
- [16] Yao Xin. Evolutionary artificial neural networks. Int. J. Neural Syst. 1993 . 4(3):203 ~ 222
- [17] Bäck T, Hammel U, and Schwefel H-P. Evolutionary computation; Comments on the history and current state. IEEE Trans. Evolutionary Computation, Apr. 1997. 1: 3 ~ 17
- [18] Whitley D, Starkweather T, and Bogart C. Genetic algorithms and neural networks: Optimizing connections and connectivity. Parallel Comput., 1990 . 14(3) : 347 ~ 361
- [19] Srinivas M, Patnaik L M. Learning neural network weights using genetic algorithms—Improving performance by search-space reduction. In: Proc. 1991 IEEE Int. Joint Conf. Neural Networks(IJCNN'91 Singapore). 3. 2331 ~ 2336
- [20] Garis H de. GenNets; Genetically programmed neural nets—Using the genetic algorithm to train neural nets whose inputs and/or outputs vary in time. In: Proc. 1991 IEEE Int. Joint Conf. Neural Networks(IJCNN'91 Singapore). 2, 1391 ~ 1396
- [21] Dominic S, Das R, Whitley D, and Anderson C. Genetic reinforcement learning for neural



- networks. In; Proc. 1991 IEEE Int. Joint Conf. Neural Networks(IJCNN'91 Seattle). 2, 71 ~ 76
- [22] Dill F A, Deer B C. An exploration of genetic algorithms for the selection of connection weights in dynamical neural networks. In; Proc. IEEE 1991 National Aerospace and Electronics Conf NAECON 1991. 3, 1111 ~ 1115
- [23] Bornholdt S, Graudenz D. General asymmetric neural networks and structure design by genetic algorithms. Neural Networks, 1992, 5(2): 327 ~ 334
- [24] Janson D J, Frenzel J F. Training product unit neural networks with genetic algorithms. IEEE Expert, May 1993, 8; 26 ~ 33
- [25] Beer R D, Gallagher J C. Evolving dynamical neural networks for adaptive behavior. Adaptive Behavior, 1(1):91 ~ 122
- [26] Yao S, Wei C J, and He Z Y. Evolving wavelet neural networks for function approximation. Electron. Lett., 1996, 32(4):360 ~ 361
- [27] Greenwood G W. Training partially recurrent neural networks using evolutionary strategies. IEEE Trans. Speech Audio Processing, Feb. 1997, 5; 192 ~ 194
- [28] Osmera P. Optimization of neural networks by genetic algorithms. Neural Network World, 1995, 5(6):965 ~ 976
- [29] Sexton R S, Dorsey R E, and Johnson J D. Toward global optimization of neural networks: A comparison of the genetic algorithm and backpropagation. Decision Support Syst., 1998, 22(2):171 ~ 185
- [30] Kamo M, Kubo T, and Iwasa Y. Neural network for female mate preference, trained by a genetic algorithm. Philosophical Trans. Roy. Soc. 1998, 353(1367):399
- [31] Yoon B, Holmes D J, and Langholz G. Efficient genetic algorithms for training layered feedforward neural networks. Inform. Sci., 1994, 76(1-2):67 ~ 85
- [32] Korning P G. Training neural networks by means of genetic algorithms working on very long chromosomes. Int. J. Neural Syst., 1995, 6(3):299 ~ 316
- [33] Dobnikar A. Evolutionary design of application-specific neural networks: A genetic approach. Neural Network World, 1995, 5(1): 41 ~ 50
- [34] Heimes F, Zalesski G, Jr. W L, and Oshima M. Traditional and evolved dynamic neural networks for aircraft simulation. In; Proc. 1997 IEEE Int. Conf. Systems, Man, and Cybernetics, Part 3 (of 5). 1995 ~ 2000
- [35] Ishigami H, Fukuda T, and Arai F. Structure optimization of fuzzy neural network by genetic algorithm. Fuzzy Sets Syst., 1995, 71(3):257 ~ 264
- [36] Fang J, Xi Y. Neural network design based on evolutionary programming. Artificial Intell. Eng., 1997, 11(2): 155 ~ 161
- [37] Yao X, Liu Y. Toward designing artificial neural networks by evolution. Appl. Math. Computation, 1998, 91(1):83 ~ 90
- [38] Liu Y and Yao X. Evolutionary design of artificial neural networks with different nodes. In; Proc. 1996 IEEE Int. Conf. Evolutionary Computation(ICEC'96). Nagoya, Japan; 670 ~ 675
- [39] Smith J M. When learning guides evolution. Nature, Oct. 1987, 329(6142): 761 ~ 762
- [40] Hinton G E, Nowlan S J. How learning can guide evolution. Complex Syst., 1987, 1(3):495 ~ 502
- [41] Bengio S, Bengio Y, Cloutier J, and Gecsei J. On the optimization of a synaptic learning rule. In;

Preprints Conf. Optimality in Artificial and Biological Neural Networks. Univ. of Texas, Dallas, Feb. 1992

- [42] Baxter J. The evolution of learning algorithms for artificial neural networks. In: Green D and Bossomaier T, ed. Complex Systems. Amsterdam, The Netherlands: IOS, 1992. 313 ~ 326
- [43] Turney P, Whitley D, and Anderson R, ed. Evolutionary Computation(Special Issue on the Baldwin Effect) . 1996. 4(3) : 213 ~ 329
- [44] Kim H B, Jung S H, Kim T G, and Park H K. Fast learning method for back-propagation neural network by evolutionary adaptation of learning rates. Neurocomput. . 1996. 11(1) : 101 ~ 106
- [45] Jacobs R A. Increased rates of convergence through learning rate adaptation. Neural Networks, 1988. 1(3) : 295 ~ 307
- [46] Yamany S M, Khiani K J, and Farag A A. Applications of neural networks and genetic algorithms in the classification of endothelial cells. Pattern Recognition Lett. . 1997. 18(11—13) : 1205 ~ 1210
- [47] Back B, Laitinen T, and Sere K. Neural networks and genetic algorithms for bankruptcy predictions. Expert Syst. Applicat. : Int. J. . 1996. 11(4) : 407 ~ 413
- [48] Dellaert F and Vandewalle J. Automatic design of cellular neural networks by means of genetic algorithms; Finding a feature detector. In: Proc. IEEE Int. Workshop Cellular Neural Networks and Their Applications, 1994. 189 ~ 194
- [49] Weller P R, Summers R, and Thompson A C. Using a genetic algorithm to evolve an optimum input set for a predictive neural network. In: Proc. Ist IEE/IEEE Int. Conf. Genetic Algorithms in Engineering Systems; Innovations and Applications(GALESIA'95). Stevenage, U. K; Inst. Elect. Eng. Conf. Pub. 414, 1995. 256 ~ 258
- [50] 陈文霞, 郑君里. 利用遗传神经网络算法的 ATM 链路容量控制. 北京: 清华大学学报, 1999, 1, 39(1). 30 ~ 33
- [51] 邢文训, 谢金星. 现代优化计算方法. 北京: 清华大学出版社, 1999

# 第 7 章 盲信号处理

## 7.1 概 述

盲信号处理 (blind signal processing, BSP) 是 20 世纪最后十年中迅速发展起来的一个研究领域。它又可以分成若干个互相关联而目标有所区别的子领域, 如盲信号分离 (blind signal separation, BSS) 以及盲解卷 (blind deconvolution)、多道盲解卷、盲均衡 (blind equalization) 等。按照所取的假设条件和研究途径不同, 可以包含独立分量分析 (independent component analysis, ICA)、因子分析 (factor analysis, FA) 和独立因子分析 (IFA) 等若干课题。BSP 的研究涉及人工神经网络、统计信号处理和信息论的有关知识, 受到这些领域研究者的重视, 它具有重要的理论和应用价值, 已成为人工神经网络的主导发展方向之一。

### 7.1.1 盲信号分离问题

设有  $N$  个未知的源信号  $S_i(t), i = 1 \sim N$  构成一个列向量  $\mathbf{S}(t) = [S_1(t), S_2(t), \dots, S_N(t)]^T$  其中  $t$  是离散时刻 取值为  $0, 1, 2, \dots$ 。设  $\mathbf{A}$  是一个  $M \times N$  维矩阵, 一般称为混合矩阵 (mixing matrix)。设  $\mathbf{X}(t) = [x_1(t), x_2(t), \dots, x_M(t)]^T$  是由  $M$  个可观察信号  $x_i(t), i = 1 \sim M$  构成的列向量, 且满足下列方程:

$$\mathbf{X}(t) = \mathbf{A}\mathbf{S}(t) \quad M \geq N \quad (7-1)$$

BSS 的命题是 对任何  $t$  根据已知的  $\mathbf{X}(t)$  在  $\mathbf{A}$  未知的条件下求未知的  $\mathbf{S}(t)$ 。这构成一个无噪声的盲分离问题。设  $\mathbf{N}(t) = [n_1(t), n_2(t), \dots, n_M(t)]^T$  是由  $M$  个白色、高斯、统计独立噪声信号  $n_i(t)$  构成的列向量 且  $\mathbf{X}(t)$  满足下列方程:

$$\mathbf{X}(t) = \mathbf{A}\mathbf{S}(t) + \mathbf{N}(t) \quad M \geq N \quad (7-2)$$

则由已知的  $\mathbf{X}(t)$  在  $\mathbf{A}$  未知时求  $\mathbf{S}(t)$  的问题是一个有噪声盲分离问题。

### 7.1.2 盲解卷和多道盲解卷 / 均衡问题

设有  $N$  个离散时间  $t$  的序列  $S_i(t), i = 1 \sim N$  构成一个  $N$  维列向量序列  $\mathbf{S}(t) = [S_1(t), S_2(t), \dots, S_N(t)]^T, t = \dots, -1, 0, 1, \dots$ 。设  $M$  维观察向量序列为  $\mathbf{X}(t) = [x_1(t), x_2(t), \dots, x_M(t)]^T, \mathbf{X}(t)$  与  $\mathbf{S}(t)$  之间的关系由下列方程描述:

$$\mathbf{X}(t) = \sum_{p=-\infty}^{+\infty} \mathbf{A}_p \mathbf{S}(t-p), \quad M \geq N \quad (7-3)$$

其中各  $A_p$  是  $M \times N$  维矩阵。若在诸  $A_p$  未知的条件下，由已知的  $X(t)$  求未知的  $S(t)$  且  $N > 1$ ，则形成多道盲解卷 / 均衡问题；若  $N = 1$ ，则形成盲解卷 / 均衡问题。

式(7-3)所表示的是一个卷积关系。若有  $A_p = 0, p < 0$  则所描述的是因果系统反之，则为非因果系统。前者的实例是通信系统，后者出现在图像处理的情况。

### 7.1.3 独立分量分析

如果按照以下的几个基本假设条件来解决 BSS 问题 则称之为 ICA。这些条件是：

(1) 各源信号  $S_i(t)$  均为 0 均值、实随机变量 各源信号之间统计独立。如果每个源信号  $S_i(t)$  的概率密度函数 简称为 pdf 为  $p_i(S_i)$  则  $S(t)$  的 pdf 为  $p_s(S)$ ，可以用下式计算：

$$p_s(S) = \prod_{i=1}^N p_i(S_i) \quad (7-4)$$

(2) 源信号数  $M$  与观察信号数  $N$  相同，即  $M = N$  这时混合阵  $A$  是一个确定且未知的  $N \times N$  维方阵。假设  $A$  是满秩的 逆矩阵  $A^{-1}$  存在。

(3) 各个  $S_i(t)$  的 pdf 中只允许有一个具有高斯分布，如果具有高斯分布的源信号个数超过一个，则各源信号是不可分的。Darmois 定理严格证明了这一结论<sup>[1]</sup>。在下述各节中将从不同角度阐明此结论。

(4) 各观察器引入的噪声很小，可以忽略不计。这时可以用式(7-1)描述源信号与观察信号之间的关系且  $M = N$ 。

(5) 关于各源信号的 pdf  $p_i(S_i)$  略有一些先验知识。这一条将在 7.2 节中详释。

ICA 的思路是设置一个  $N \times N$  维反混合阵  $W = (w_{ij})$ ， $X(t)$  经过  $W$  变换后得到  $N$  维输出列向量  $Y(t) = [y_1(t), y_2(t), \dots, y_N(t)]^T$  即有

$$Y(t) = WX(t) = WAS(t) \quad (7-5)$$

如果通过学习得以实现  $WA = I$  ( $I$  是  $N \times N$  维单位阵) 则  $Y(t) = S(t)$ ，从而达到了源信号分离目标。由于没有任何参照目标，这一学习只能是自组织的。学习过程的第一步是建立一个以  $W$  为变元的目标函数  $L(W)$  如果某个  $\hat{W}$  能使  $L(W)$  达到极大(小)值，该  $\hat{W}$  即为所需的解。第二步即是用一种有效的算法求  $W$ 。按照  $L(W)$  定义的不同和求  $W$  的方法不同可以构成各种 ICA 算法。令人吃惊的是这些算法都出奇地简单，而且，即使关于源信号 pdf 的先验知识很不充分，分离效果也出奇地好。本章主要内容即是讨论各种 ICA 算法及其实验结果。

### 7.1.4 因子分析和独立因子分析

FA 采用式(7-2)作为由源信号得到观察信号的计算公式。FA 算法的基本假设是：

(1) 各个源信号  $S_i(t)$  为 0 均值、高斯随机变量，各个源信号之间不相关，即满足

$$E[S_i(t)S_j(t)] = 0 \quad 1 \leq i < j \leq N \quad (7-6)$$

其中  $E[\cdot]$  表示对集合取平均。

(2) 允许  $N \neq M$ 。

(3) 存在噪声 即  $N(t) \neq 0$ 。

由于 FA 只能利用各  $x_i(t)$  之间的二阶统计信息，因此不可能实现由观察信号对源信号的正确分离和恢复，在没有噪声的条件下 FA 将退化为只利用二阶统计（相关）信息的 PCA。7.2 节将详细讨论这一点。

ICA 算法由于能利用高于二阶的统计信息而正确实现源信号分离和恢复。但是其假设条件较严，特别是没有噪声和  $M = N$  这两条往往与实际情况不符。当噪声较大时，源信号的分离效果显著恶化。IFA 算法将 FA 的第一条假设修改为各  $S_i(t)$  具有非高斯分布特性且相互统计独立，第二、三条假设保持不变，从而兼取 ICA 算法和 FA 算法的优点。IFA 算法的构成思路与 FA 算法无异，究竟实用效果如何尚待进一步研究。7.10 节和 7.11 节将介绍 FA 和 IFA 算法。

### 7.1.5 BSP 的应用

#### (1) 鸡尾酒会问题

在鸡尾酒会上有多人同时说话，如何将这些话语分辨出来成为 BSS 的第一批标准研究课题。A. J. Bell 等对此问题的研究成为 ICA 算法发展史的里程碑之一<sup>[2]</sup>。该项研究能分辨出 10 个说话人的话语，显示 ICA 的巨大潜力。

#### (2) 医学信号处理

包括心电图 (ECG)<sup>[3]</sup>、脑电图 (EEG)<sup>[4,5]</sup> 等信号的分离。其中尤其是 EEG 中混有各种源信号，BSS 的应用有很大潜在价值。

#### (3) 天线阵列处理

雷达和声纳系统中的天线阵列处理，实现最佳的源定位和干扰抑制<sup>[6]</sup>以及各种阵列信号处理。<sup>[6,7]</sup>

#### (4) 通信系统中的各种应用

特别是移动通信系统中的盲解卷 / 均衡问题<sup>[7~10]</sup>

#### (5) 语音增强<sup>[5]</sup>

#### (6) 图像恢复、视觉信号处理<sup>[11,2,12]</sup>

#### (7) 回波抵消、交混回响消除<sup>[2]</sup>

#### (8) 地球物理信号处理

包括地震信号盲解卷、探油信号处理等<sup>[2]</sup>

#### (9) 系统的盲辨识<sup>[6]</sup>

#### (10) 财政金融领域中的预测<sup>[13]</sup>

#### (11) 统计信号处理中的贝叶斯检测、数据分析和压缩<sup>[6]</sup>

#### (12) 混沌 chaos 信号分析<sup>[6,14]</sup>

### 7.1.6 ICA 的发展简史

C. Jutten 和 J. Herault 于 1991 年首创将人工神经网络算法用于 BSS 问题<sup>[15]</sup>，从而开启了一个新领域。虽然他们的学习算法是启发式的并且没有明确指出需利用观察信号的高阶（高于二阶）统计信息，但是其迭代计算公式已具备后来算法的雏形。

1994 年, P. Comon 首先界定了解决 BSS 问题的 ICA 方法的基本假设条件 (ICA 这个名称就是他提出的)。他还明确指出, 应该通过使某个称为对比函数 (contrast function) 的目标函数达到极大值来消除观察信号中高阶统计关联, 从而实现 BSS<sup>[6]</sup>。他还指出 ICA 是 PCA 的扩展和推广。

1995 年 A. J. Bell 和 T. J. Sejnowski 发表了 ICA 发展史中的里程碑文献<sup>[2]</sup>。其重要贡献在于: 第一, 利用神经网络的非线性特性来消除观察信号中的高阶统计关联 (文中采用的是具有 Sigmoid 函数的神经元, 因此只适用于具有超高斯 pdf 的信号 (例如语音的分离问题, 超高斯的概念见 5.2 节))。第二, 用信息最大化准则建立目标函数, 从而将信息论方法与 ICA 结合起来。第三 给出了神经网络式的最优  $\mathbf{w}$  迭代学习算法, 成为后续各种算法的基础。第四 成功地对具有 10 个说话人的鸡尾酒会问题给出了很好的分离效果。因为证明了 ICA 是一种解决 BSS 问题的简单、高效算法, 从而带起了一大批后续的研究工作。

1996 年, B. A. Pearlmutter 在 ICA 中引入以最大似然为准则的目标函数<sup>[12]</sup>。同年, J-F. Cardoso 和 B. H. Laheld 提出了 ICA 学习算法中的“相对梯度”、“等价变化”和有关稳定性和分离精度等重要思路和方法<sup>[16]</sup>。

1997 年, D. T. Pham 和 P. Garat 通过准最大似然途径对 ICA 的学习算法、稳定性、分离精度和源 pdf 的确定作了进一步讨论<sup>[17]</sup>。

1998 年, PIEEE 的 10 月号的论文集为 BSP 专集, 其中文献[8]和[18]对这一领域的成果作了综述并指出进一步发展方向。

1999 年, 获得了更多的研究成果, 例如关于如何确定源信号 pdf 的问题<sup>[5,19]</sup>。

### 7.1.7 ICA 研究中的主要问题

(1) 关于源信号不同 pdf 如高斯、超高斯、次高斯 pdf 的特性描述。给出不同特性 pdf 随机变量的特征参数: 矩 (moment)、累积矩 (cumulant) 和峰起度 (kurtosis)。研究解的等价性, 去除二阶相关及去除高阶相关等问题。以上内容将在 7.2 节讨论。

(2) ICA 的目标函数。7.3 节将讨论如何选择目标函数, 所取的准则包括互信息与信息最大化准则、最大似然准则、最大独立性准则、白化 (去除二阶相关) 后去除高阶相关准则、在白化约束下去除高阶相关准则以及其他启发式准则。

(3) ICA 的学习算法。与其他神经网络学习算法相同, 学习可取批处理方式 (针对平稳环境) 或在线自适应方式 (针对变化或平稳环境)。为了求得使目标函数达到极值的  $\mathbf{w}$  所用的算法一般为迭代算法, 应尽量简单、收敛快。7.4 节将讨论相对梯度和自然梯度等算法, 它们从不同的视点出发导出了同样非常简洁、高效的迭代计算公式。这些算法还具有“等价变化”的优良性能 (见 7.4 节)。

(4) ICA 算法的稳定性。稳定性是指 ICA 迭代计算中达到正确源信号分离的解可能是一个平衡点而不是稳定点。7.5 节将讨论此问题。

(5) ICA 算法实现的源信号分离精度, 即相邻源信号的干扰问题, 这在 7.6 节讨论。

(6) ICA 中源信号 pdf 的确定。如果关于源信号 pdf 的先验知识很少甚至完全没有时, 必须在学习过程中加以确定, 否则将进行反复尝试, 可能会浪费大量时间而且使分离效果不佳。这将在 7.7 节讨论。

(7) ICA 的各种模拟实验和具体应用研究, 在 7.8 节中讨论。

(8) 盲解卷 / 均衡和多道盲解卷 / 均衡问题, 在 7.9 节讨论。

(9) 有噪声和  $M \neq N$  的情况下, 将主要研究 FA 和 IFA 方案 这放在 7.10、7.11 节中讨论。有关 ICA 进一步的研究课题在 7.12 节中列出。

## 7.2 源信号 pdf 描述、等价可分性、二阶和高阶相关的去除

### 7.2.1 源信号 pdf 描述

按照 ICA 的假设条件, 各源信号统计独立, 因此每一源信号  $S_i$  可用其 pdf 来描述 表示为  $p_i(S_i)$ 。 $S_i$  作为随机变量, 其基本统计特征参数是它的各阶矩  $E[S_i^l]$  ( $l = 1, 2, 3, 4, \dots$  为阶数),

$$E[S_i^l] = \int S_i^l p_i(S_i) dS_i \quad (7-7)$$

在 ICA 中 设各  $S_i$  的均值为 0 且其 pdf 具有对称分布, 这时它的一阶矩 (即均值  $m_i$ )、三阶矩、五阶矩等皆为 0; 这时只需考虑二阶矩  $E[S_i^2]$  (即均方差值时) 和四阶矩  $E[S_i^4]$  更高阶矩往往不予考虑。在上述假设下  $S_i$  的另一种重要统计特征参数是它的二阶和四阶累积矩 (cumulant)  $K_2$  和  $K_4$  其定义为

$$\left. \begin{aligned} K_2 &= E[S_i^2] \\ K_4 &= E[S_i^4] - 3(E[S_i^2])^2 \end{aligned} \right\} \quad (7-8)$$

由此可以产生关于  $S_i$  的 pdf 特性的一个重要参数, 即峰起度 kurtosis  $\mathcal{K}_i$  其定义为

$$\mathcal{K}_i = \frac{K_4}{K_2^2} = E[S_i^4] - 3(E[S_i^2])^2 \quad (7-9)$$

$\mathcal{K}_i$  取决于  $p_i(S_i)$  的特性, 可以分成以下三种情况。

(1) 如果  $p_i(S_i)$  为高斯函数 则  $\mathcal{K}_i = 0$ 。有许多噪声具有此特性。当  $S_i$  为多个随机变量之和且变量数很多时,  $p_i(S_i)$  也趋近于高斯分布函数。高斯函数如图 7-1 中曲线 ① 所示。

(2) 如果  $p_i(S_i)$  为超高斯 (supergaussian) 函数 则  $\mathcal{K}_i > 0$ 。这种分布函数中心部分又窄又高 而尾部拖得很长 如图 7-1 中曲线 ② 所示。自然界的语音和某些音乐信号具此特性。

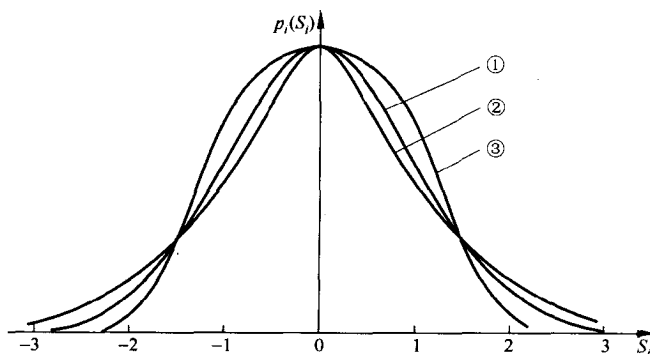


图 7-1 高斯、超高斯、次高斯 pdf

(3) 如果  $p_i(S_i)$  为次高斯 (subgaussian) 函数, 则  $\mathcal{K}_i < 0$ 。这种分布函数中心部分较宽, 尾部很短, 如图 7-1 中曲线 ③ 所示。自然界中的图像信号具有此特性。此外, 在一定区间中具有均匀分布 (即在此区间中 pdf 值等于常数) 的函数, 也属于此情况。

如果事先知道各信号源的 pdf 均为超高斯的或均为次高斯的, 则可以用 ICA 实现效果优良的信号分离 (即使不知道各 pdf 的准确特性)。反之, 如果不知道各 pdf 属何种特性 (超高斯或次高斯) 或为二者都存在的情况, 则必须采取措施在学习过程中予以确定。

### 7.2.2 源信号与混合阵的等价尺度变换

由于观察信号  $\mathbf{X}(t)$  是源信号  $\mathbf{S}(t)$  与混合阵  $\mathbf{A}$  之积 (式 7-1)) 由  $\mathbf{X}(t)$  无法确定后二者的尺度值。这由下列式 (7-1) 的展开式可以看到:

$$x_i(t) = \sum_{j=1}^N a_{ij} S_j(t) = \sum_{j=1}^N (\eta_j^{-1} a_{ij}) (\eta_j S_j(t)), \quad i = 1 \sim N$$

其中  $a_{ij}$  是  $\mathbf{A}$  的第  $i$  行、 $j$  列元素。若  $S_j(t)$  乘以任何非 0 系数  $\eta_j$ , 而  $\mathbf{A}$  的第  $j$  列各元素皆乘以  $\eta_j^{-1}$ , 则不管各  $\eta_j$  取何值  $x_i(t)$  不变。因此由  $\mathbf{X}(t)$  试图获得各源信号时存在尺度不确定性。为消除这一点, ICA 约定各源信号的均方差值为 1, 即  $E[S_i^2] = 1, \forall i$ 。这时  $\mathbf{S}(t)$  的相关阵是一个单位阵, 即

$$E[\mathbf{S}(t)\mathbf{S}^T(t)] = \mathbf{I} \quad (7-10)$$

### 7.2.3 解的等价性

ICA 的目标是求一个变换阵, 或称分离阵  $\mathbf{W}$  使得  $\mathbf{Y}(t) = \mathbf{W}\mathbf{X}(t)$  与  $\mathbf{S}(t)$  对应。这种对应并不意味着二者的排序一一对应 (即  $y_1(t)$  与  $S_1(t)$  对应,  $y_2(t)$  与  $S_2(t)$  对应, 等等), 也并不意味着各  $y_i(t)$  与各  $S_i(t)$  的尺度完全一致 (其原因如上一小节所述)。如果按这种理解, ICA 就会有許多等价解。设  $\hat{\mathbf{W}}$  是一个解, 那么  $\mathbf{P}\mathbf{A}\hat{\mathbf{W}}$  也是一个解, 其中  $\mathbf{A}$  是一个对角阵, 即其对角元素取任何非 0 值, 其他元素为 0, 即

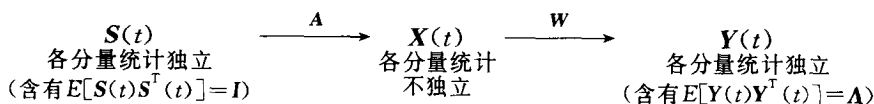
$$\mathbf{A} = \text{diag}[\Lambda_{11}, \Lambda_{22}, \dots, \Lambda_{NN}] \quad (7-11)$$

$\mathbf{P}$  是一个换位阵 (permutation matrix), 它的每行、每列中只有一个元素为 1, 其他元素为 0。下面以一个  $3 \times 3$  维的  $\mathbf{P}$  为例, 来表明其换位作用,

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

可以看到, 经过此换位阵变换后,  $y_1 = x_2, y_2 = x_1, y_3 = x_3$ 。

可以看到, ICA 实现正确信号分离的结果只能表现在  $\mathbf{Y}(t)$  的各个分量统计独立上, 这时  $\mathbf{Y}(t)$  的自相关阵当然是对角阵, 即  $E[\mathbf{Y}(t)\mathbf{Y}^T(t)] = \mathbf{A}$ 。这样, ICA 的约定条件和待解决问题可由下列流图表明:





## 7.2.4 去除二阶相关和去除高阶相关

### 1. 旋转阵与二阶相关性、高阶相关性

(1) 一个  $N \times N$  维矩阵  $\mathbf{A}$  满足条件

$$\mathbf{A}\mathbf{A}^T = \mathbf{I} \quad \text{或} \quad \mathbf{A}^T = \mathbf{A}^{-1}$$

时即是旋转阵。如前所述, 对角阵  $\mathbf{A}$  是一个尺度变换阵 (式 7-11))。设有两个  $N \times N$  维对角阵  $\mathbf{A}_a, \mathbf{A}_b, \mathbf{A}\mathbf{A}_a\mathbf{A}^T = \mathbf{A}_b$ 。设  $\tilde{\mathbf{A}} = \mathbf{A}_b^{-\frac{1}{2}}\mathbf{A}\mathbf{A}_a^{\frac{1}{2}}$  则有  $\tilde{\mathbf{A}}\tilde{\mathbf{A}}^T = \mathbf{I}, \mathbf{A} = \mathbf{A}_b^{\frac{1}{2}}\tilde{\mathbf{A}}\mathbf{A}_a^{-\frac{1}{2}}$ 。这时  $\mathbf{A}$  是旋转阵, 而  $\mathbf{A}$  是旋转-尺度变换阵。

(2) 若一个 ICA 问题中的混合阵  $\mathbf{A}$  是旋转阵或旋转-尺度变换阵, 则观察向量  $\mathbf{X}(t)$  的相关阵  $\mathbf{R}_{xx}$  是单位阵或对角阵, 这可简单地导出, 即

$$\begin{aligned} \mathbf{R}_{xx} &= E[\mathbf{X}(t)\mathbf{X}^T(t)] = E[\mathbf{A}\mathbf{S}(t)(\mathbf{A}\mathbf{S}(t))^T] = E[\mathbf{A}\mathbf{S}(t)\mathbf{S}^T(t)\mathbf{A}^T] \\ &= \mathbf{A}E[\mathbf{S}(t)\mathbf{S}^T(t)]\mathbf{A}^T \end{aligned}$$

根据 ICA 假设条件  $\mathbf{R}_{ss} = E[\mathbf{S}(t)\mathbf{S}^T(t)] = \mathbf{I}$ 。再根据旋转或旋转-尺度变换阵的定义  $\mathbf{R}_{xx}$  为单位阵或对角阵。这时  $\mathbf{X}(t)$  的各分量之间不存在二阶相关性, 但是存在高阶相关性。

(3) 若混合阵非旋转阵或旋转-尺度变换阵, 则  $\mathbf{R}_{xx}$  不是单位阵或对角阵, 这时  $\mathbf{X}(t)$  各分量之间既存在二阶相关性又存在高阶相关性。

### 2. 相关性的去除

一般  $\mathbf{X}(t)$  各分量之间既包含二阶又包含高阶相关性, 如通过乘  $\mathbf{W}$  得到的  $\mathbf{Y}(t)$  各分量之间不包含任何相关性, 从而可实现源信号分离。ICA 所取的一般方案是用一种迭代算法统一消除所有相关性, 使  $\mathbf{Y}(t)$  各分量统计独立。还有一种途径是首先消除二阶相关性 (称为白化), 然后消除高阶相关性, 或在白化约束下消除高阶相关性。为了阐明通过消除统计相关性以实现 BSS 的机理, 此处讨论先白化后除高阶相关的途径。

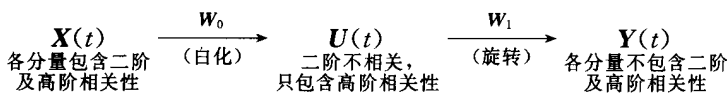
设有一个  $N$  维中间列向量  $\mathbf{U}(t)$  以及两个  $N \times N$  维矩阵  $\mathbf{W}_0$  和  $\mathbf{W}_1, \mathbf{U}(t) = \mathbf{W}_0\mathbf{X}(t), \mathbf{Y}(t) = \mathbf{W}_1\mathbf{U}(t)$ 。 $\mathbf{W}_0$  称为白化阵, 其作用是使  $\mathbf{U}(t)$  各分量之间不存在二阶相关性, 即实现

$$\mathbf{R}_{uu} = E[\mathbf{U}(t)\mathbf{U}^T(t)] = \mathbf{W}_0\mathbf{R}_{xx}\mathbf{W}_0^T = \mathbf{A}_0$$

其中  $\mathbf{A}_0$  为对角阵。这一步通过乘  $\mathbf{W}_0$  将具有非对角相关阵  $\mathbf{R}_{xx}$  的向量  $\mathbf{X}(t)$  变换成了具有对角相关阵  $\mathbf{R}_{uu} = \mathbf{A}_0$  的向量  $\mathbf{U}(t)$ 。可以看到这一步做的是 PCA。第二步在保持二阶不相关的条件下去除高阶相关, 这时必然有

$$\mathbf{R}_{yy} = E[\mathbf{Y}(t)\mathbf{Y}^T(t)] = \mathbf{W}_1\mathbf{R}_{uu}\mathbf{W}_1^T = \mathbf{W}_1\mathbf{A}_0\mathbf{W}_1^T = \mathbf{A}_1$$

其中  $\mathbf{A}_1$  为对角阵。这说明  $\mathbf{W}_1$  必须是旋转阵或旋转-尺度变换阵。这两步过程可由下列流程图显示:



这样整个源信号分离过程可表示为

$$\mathbf{Y}(t) = \mathbf{W}_1\mathbf{U}(t) = \mathbf{W}_1\mathbf{W}_0\mathbf{X}(t) = \mathbf{W}\mathbf{X}(t)$$

其中

$$\mathbf{W} = \mathbf{W}_1\mathbf{W}_0$$

可以看到, 实现 BSS 等效于求得混合阵  $\mathbf{A}$  也就是要确定  $N^2$  个待定参数。在上述第一

步白化过程中，只有  $N(N-1)/2$  个约束方程，即

$$E[u_i(t)u_j(t)] = 0, \quad 1 \leq i < j \leq N$$

这时只能确定  $N(N-1)/2$  个参数 因此剩余的  $N(N+1)/2$  个参数必须在第二步确定。

### 3. 信号源不可分的原因

为什么信号源的 pdf 中有两个或更多个为高斯函数时，这些信号源是不可分的？

设  $N$  个信号源皆具有 0 均值、单位均方差值高斯分布且相互统计独立，则  $\mathbf{S}$  的 pdf 为

$$p(\mathbf{S}) = \prod_{i=1}^N p_i(S_i) = \prod_{i=1}^N (2\pi)^{-\frac{1}{2}} \exp\left(-\frac{S_i^2}{2}\right) = (2\pi)^{-\frac{N}{2}} \exp\left(-\frac{\mathbf{S}^T \mathbf{S}}{2}\right)$$

由  $\mathbf{X} = \mathbf{A}\mathbf{S}$  可知  $\mathbf{X}$  的 pdf 为(见 7.3.1 节)

$$p(\mathbf{X}) = |\det \mathbf{A}|^{-1} p(\mathbf{S})|_{\mathbf{s}=\mathbf{A}^{-1}\mathbf{x}} = |\det \mathbf{A}|^{-1} (2\pi)^{-\frac{N}{2}} \exp\left(-\frac{\mathbf{X}^T \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{X}}{2}\right)$$

其中  $\det \mathbf{A}$  表示  $\mathbf{A}$  的行列式值。已知  $\mathbf{S}$  的协方差阵为  $\mathbf{R}_{\mathbf{ss}} = \mathbf{I}$ ，则  $\mathbf{X}$  的协方差阵为  $\mathbf{R}_{\mathbf{xx}} = \mathbf{A}\mathbf{R}_{\mathbf{ss}}\mathbf{A}^T = \mathbf{A}\mathbf{A}^T$ ， $\mathbf{R}_{\mathbf{xx}}^{-1} = \mathbf{A}^{-T}\mathbf{A}^{-1}$ ，即有

$$p(\mathbf{X}) = |\det \mathbf{A}|^{-1} (2\pi)^{-\frac{N}{2}} \exp\left(-\frac{\mathbf{X}^T \mathbf{R}_{\mathbf{xx}}^{-1} \mathbf{X}}{2}\right)$$

设  $\mathbf{U} = \mathbf{W}_0 \mathbf{X}$ ， $\mathbf{W}_0$  是一个白化阵 则  $\mathbf{R}_{\mathbf{uu}} = \mathbf{W}_0 \mathbf{R}_{\mathbf{xx}} \mathbf{W}_0^T = \mathbf{A}_0 = \text{diag}[\Lambda_{011}, \Lambda_{022}, \dots, \Lambda_{0NN}]$  易证明  $\mathbf{U}$  的 pdf 为

$$\begin{aligned} p(\mathbf{U}) &= |\det \mathbf{W}_0|^{-1} |\det \mathbf{A}|^{-1} (2\pi)^{-\frac{N}{2}} \exp\left(-\frac{\mathbf{U}^T \mathbf{A}_0^{-1} \mathbf{U}}{2}\right) \\ &= |\det \mathbf{A}|^{-1} \prod_{i=1}^N \Lambda_{0ii}^{-\frac{1}{2}} (2\pi)^{-\frac{1}{2}} \exp\left(-\frac{u_i^2}{2\Lambda_{0ii}}\right) \end{aligned}$$

可以看到，这时  $\mathbf{U}$  的各个分量是统计独立的，但是信号分离的目标并未达到，这可由下列分析表明。

设  $\mathbf{Y} = \mathbf{W}_1 \mathbf{X}$ ， $\mathbf{W}_1$  是一个任意的旋转阵 则  $\mathbf{R}_{\mathbf{yy}} = \mathbf{W}_1 \mathbf{A}_0 \mathbf{W}_1^T = \mathbf{A}_1 = \text{diag}[\Lambda_{111}, \Lambda_{122}, \dots, \Lambda_{1NN}]$ ，因而可同样证明  $\mathbf{Y}$  的 pdf 为 ( $C$  为一常数)

$$p(\mathbf{Y}) = C \prod_{i=1}^N \Lambda_{1ii}^{-\frac{1}{2}} (2\pi)^{-\frac{1}{2}} \exp\left(-\frac{y_i^2}{2\Lambda_{1ii}}\right)$$

这说明，无论取什么样的旋转阵  $\mathbf{W}_1$  所得到的  $\mathbf{Y}$  都是统计独立的。这就是说，当源信号为高斯分布随机变量时 其 pdf 只涉及二阶统计特性。由于没有高阶统计特性可资利用，而任何旋转变化又不会改变向量的二阶不相关性。因此，由任意旋转得到的结果都符合统计独立要求，而它们显然与源信号不可能总是一致。

## 7.3 ICA 的目标函数

如 7.1.3 小节所述，ICA 的目标是通过变换  $\mathbf{Y}(t) = \mathbf{W}\mathbf{X}(t)$  (式 7-5)) 由观察信号向量求得源信号向量。为此需设置一个目标函数  $L(\mathbf{W})$  如  $\hat{\mathbf{W}}$  能使之达到极大(小)值 则此  $\mathbf{W}$  即为所需解 即使得  $\mathbf{Y}(t)$  与  $\mathbf{S}(t)$  相对应。 $L(\mathbf{W})$  应简洁、易计算。这一节将给出有关的预备知识，并且从不同的视角出发导致相同的目标函数。

### 7.3.1 若干预备知识

#### 1. 两个 pdf 的比较, Kullback-Leibler 发散度

设  $\mathbf{X}$  为  $N$  维列向量,  $p_a(\mathbf{X})$  和  $p_b(\mathbf{X})$  是两个 pdf, 它们之间的相似程度可以用 Kullback-Leibler 发散度 简记为  $KL$  发散度 来衡量。其定义如下 其中  $\ln$  表示自然对数):

$$KL[p_a(\mathbf{X}) | p_b(\mathbf{X})] = \int_{\mathbf{X}} p_a(\mathbf{X}) \ln \left[ \frac{p_a(\mathbf{X})}{p_b(\mathbf{X})} \right] d\mathbf{X} \quad (7-12)$$

$KL$  发散度的主要性质是

$$KL[p_a(\mathbf{X}) | p_b(\mathbf{X})] \geq 0 \quad (7-13)$$

二者越不相似,  $KL$  值越大。当且仅当  $p_a(\mathbf{X}) = p_b(\mathbf{X})$  时, 式(7-13) 取等号。这一性质的证明及  $KL$  发散度的其他重要性质可参见文献[16]、[20] 为简洁计不再赘述。

#### 2. 随机向量各分量统计独立性的衡量

设  $N$  维随机列向量  $\mathbf{X}$  的 pdf 为  $p_{\mathbf{X}}(\mathbf{X})$  它的各个分量  $x_i$  的 pdf 为  $p_i(x_i), i = 1 \sim N$ 。

可以用  $p_{\mathbf{X}}(\mathbf{X})$  和  $\prod p_i(x_i)$  之间的  $KL$  发散度来衡量  $\mathbf{X}$  各分量之间的统计独立性。这一量也称为  $\mathbf{X}$  各分量间的互信息 并表示为  $I(\mathbf{X})$  即有

$$I(\mathbf{X}) = KL[p_{\mathbf{X}}(\mathbf{X}) | \prod_{i=1}^N p_i(x_i)] = \int_{\mathbf{X}} p_{\mathbf{X}}(\mathbf{X}) \ln \left[ \frac{p_{\mathbf{X}}(\mathbf{X})}{\prod_{i=1}^N p_i(x_i)} \right] d\mathbf{X} \quad (7-14)$$

可以看到,  $I(\mathbf{X}) = 0, p_{\mathbf{X}}(\mathbf{X}) = \prod_{i=1}^N p_i(x_i), \mathbf{X}$  的各分量统计独立, 这三种表述完全等价。

ICA 的思路是通过变换, 使  $\mathbf{Y}(t)$  各分量统计独立来实现源信号分离,  $I(\mathbf{X})$  显然可作为一种目标函数。这是 P. Comon 首先提出的<sup>[6]</sup>, 他将其称为对比函数 (contrast function) 所以后来的一些文献中也沿用这一名称。

#### 3. 线性变换下两个 pdf 之间的关系

设  $\mathbf{X}$  为  $N$  维随机向量 其 pdf 为  $p_{\mathbf{X}}(\mathbf{X})$ 。  $\mathbf{Y} = \mathbf{W}\mathbf{X}$ ,  $\mathbf{W}$  为满秩  $N \times N$  维矩阵。这样  $\mathbf{Y}$  也是  $N$  维随机向量 其 pdf 为  $p_{\mathbf{Y}}(\mathbf{Y})$ 。这两个随机向量的 pdf 之间满足下列关系 (证明从略):

$$p_{\mathbf{X}}(\mathbf{X}) = |\det \mathbf{W}| p_{\mathbf{Y}}(\mathbf{Y}) |_{\mathbf{Y}=\mathbf{W}\mathbf{X}} \quad (7-15)$$

或

$$p_{\mathbf{Y}}(\mathbf{Y}) = |\det \mathbf{W}|^{-1} p_{\mathbf{X}}(\mathbf{X}) |_{\mathbf{X}=\mathbf{W}^{-1}\mathbf{Y}} \quad (7-16)$$

其中  $\det \mathbf{W}$  是  $\mathbf{W}$  的行列式, 注意下式成立:

$$|\det \mathbf{W}| = |\det(\mathbf{W}\mathbf{W}^T)|^{\frac{1}{2}} \quad (7-17)$$

有些文献即用此式右侧代入式(7-15)和式(7-16)中。

#### 4. 非线性变换下两个 pdf 之间的关系

设两个  $N$  维随机向量  $\mathbf{Y}$  和  $\mathbf{X}$  之间具有非线性变换关系  $\mathbf{Y} = \mathbf{G}(\mathbf{X})$  或写成  $y_i = g_i(\mathbf{X})$ ,  $i = 1 \sim N$  其中各  $g_i(\cdot)$  为非线性函数 则二者 pdf 即  $p_{\mathbf{Y}}(\mathbf{Y})$  与  $p_{\mathbf{X}}(\mathbf{X})$  之间具有下列关系:

$$p_{\mathbf{X}}(\mathbf{X}) = |\det \mathbf{J}(\mathbf{G})| p_{\mathbf{Y}}(\mathbf{Y}) |_{\mathbf{Y}=\mathbf{G}(\mathbf{X})} \quad (7-18)$$

$J(G)$  表示  $G(X)$  的雅可比 (Jacobian) 阵, 其第  $i$  行  $j$  列元素  $J_{ij}$  按下式计算:

$$J_{ij} = \frac{\partial y_i}{\partial x_j} = \frac{\partial g_i(\mathbf{X})}{\partial x_j}, \quad i = 1 \sim N, j = 1 \sim N \quad (7-19)$$

如果  $J(G)$  为满秩, 则有

$$p_Y(\mathbf{Y}) = |\det J(G)|^{-1} p_X(\mathbf{X}) \Big|_{\mathbf{X}=G^{-1}(\mathbf{Y})} \quad (7-20)$$

一种简单的非线性变换是设有一个中间  $N$  维列向量  $\mathbf{Z} = [z_1, z_2, \dots, z_N]^T = \mathbf{W}\mathbf{X}$  且  $y_i = g_i(z_i), i = 1 \sim N$  其中  $g_i(\cdot)$  是某种非线性函数。这时式 (7-19) 将具有下列形式

(注意:  $\mathbf{W} = (w_{ij}), z_i = \sum_{j=1}^N w_{ij} x_j$ ):

$$J_{ij} = \frac{\partial y_i}{\partial x_j} = \frac{dg_i(z_i)}{dz_i} \cdot \frac{\partial z_i}{\partial x_j} = \frac{dg_i(z_i)}{dz_i} w_{ij}, \quad i = 1 \sim N, j = 1 \sim N$$

若各  $g_i(\cdot)$  为线性函数, 则  $J(G)$  退化为  $\mathbf{W}$ 。

### 7.3.2 最大似然目标函数

7.1.3 小节指出, 如能求出反混合阵  $\mathbf{W}$  使得  $\mathbf{W}\mathbf{A} = \mathbf{I}$  则可求得  $\mathbf{Y}(t) = \mathbf{S}(t)$  (见式 (7-5)) 实现源信号分离。由此得  $\mathbf{A} = \mathbf{W}^{-1}$  这就是说, 求反混合阵  $\mathbf{W}$  等效于求混合阵  $\mathbf{A}$ 。这样, 可以将  $\mathbf{S}(t)$  至  $\mathbf{X}(t)$  的变换方程 (式 7-1)) 改写为

$$\mathbf{X}(t) = \mathbf{A}\mathbf{S}(t) = \mathbf{W}^{-1}\mathbf{S}(t) \quad (7-22)$$

其中  $\mathbf{W}$  待定。已知  $\mathbf{S}(t)$  的 pdf 为  $p_S(\mathbf{S}) = \prod p_i(S_i)$  (见式 7-4))。按照式 (7-22),  $\mathbf{S}(t) = \mathbf{W}\mathbf{X}(t)$ , 则可根据式 7-15 得到  $\mathbf{X}(t)$  的似然 pdf  $\hat{p}_X(\mathbf{X})$  即有

$$\hat{p}_X(\mathbf{X}) = |\det \mathbf{W}| p_S(\mathbf{S}) \Big|_{\mathbf{S}=\mathbf{W}\mathbf{X}} = |\det \mathbf{W}| \prod_{i=1}^N p_i(S_i) \Big|_{\mathbf{S}=\mathbf{W}\mathbf{X}} \quad (7-23)$$

由于此式右侧的  $\mathbf{W}$  是待定的,  $\hat{p}_X(\mathbf{X})$  也是  $\mathbf{W}$  的函数, 可以将其明显地表示为  $\hat{p}_X(\mathbf{X}, \mathbf{W})$  其自然对数用  $l(\mathbf{X}, \mathbf{W})$  表示, 称为随机向量  $\mathbf{X}$  的似然值 (likelihood), 即

$$l(\mathbf{X}, \mathbf{W}) = \ln \hat{p}_X(\mathbf{X}, \mathbf{W}) = \ln |\det \mathbf{W}| + \sum_{i=1}^N \ln p_i(S_i) \Big|_{\mathbf{S}=\mathbf{W}\mathbf{X}} \quad (7-24)$$

因此如能求得一个  $\hat{\mathbf{W}}$  使得  $l(\mathbf{X}, \mathbf{W})$  对于  $\mathbf{X}$  的集合平均值达到最大值, 则  $\hat{\mathbf{W}}$  即是所需的解。这称为最大似然 (maximum likelihood, ML) 原理。这样, 可以将  $l(\mathbf{X}, \mathbf{W})$  的集合平均值  $E[l(\mathbf{X}, \mathbf{W})]$  定为 ICA 的目标函数, 记为  $L_{ML}(\mathbf{W})$  即

$$L_{ML}(\mathbf{W}) = E[l(\mathbf{X}, \mathbf{W})] = \int_{\mathbf{X}} p_X(\mathbf{X}) \ln \hat{p}_X(\mathbf{X}, \mathbf{W}) d\mathbf{X} \quad (7-25)$$

注意: 式中的  $p_X(\mathbf{X})$  为  $\mathbf{X}$  的真实 pdf。若有  $\hat{\mathbf{W}}$  使  $L_{ML}(\mathbf{W})$  最大, 即为 ICA 的解。

另外一种方案, 是将目标函数定义为  $p_X(\mathbf{X})$  和  $p_X(\mathbf{X}, \mathbf{W})$  之间的 KL 发散度, 记为  $L_{KL}(\mathbf{W})$ , 即

$$L_{KL}(\mathbf{W}) = KL(p_X(\mathbf{X}) \mid \hat{p}_X(\mathbf{X}, \mathbf{W})) = \int_{\mathbf{X}} p_X(\mathbf{X}) \ln \left[ \frac{p_X(\mathbf{X})}{\hat{p}_X(\mathbf{X}, \mathbf{W})} \right] d\mathbf{X} \quad (7-26)$$

注意到随机向量  $\mathbf{X}$  的熵  $H(\mathbf{X})$  为

$$H(\mathbf{X}) = - \int_{\mathbf{X}} p_{\mathbf{X}}(\mathbf{X}) \ln p_{\mathbf{X}}(\mathbf{X}) d\mathbf{X} \quad (7-27)$$

且与  $\mathbf{W}$  无关 因此  $L_{\text{ML}}(\mathbf{W})$  和  $L_{\text{KL}}(\mathbf{W})$  之间满足下列关系：

$$L_{\text{KL}}(\mathbf{W}) = -H(\mathbf{X}) - L_{\text{ML}}(\mathbf{W}) \quad (7-28)$$

这样 当  $\mathbf{W}$  使  $L_{\text{ML}}(\mathbf{W})$  达到极大值时 将使  $L_{\text{KL}}(\mathbf{W})$  达到极小值。所以取 KL 发散度为目标函数时，使其达极小值的  $\mathbf{W}$  是 ICA 的解 ( $H(\mathbf{X})$  与  $\mathbf{W}$  无关 可作为常数)

如观察向量的个数有限，可记为  $\mathbf{X}(1), \mathbf{X}(2), \dots, \mathbf{X}(T)$  则式 (7-25) 的集合平均只能用有限多个样本的平均来近似取代，这时目标函数记为  $L_{\text{ML}}(\mathbf{W})$  其计算公式为

$$\tilde{L}_{\text{ML}}(\mathbf{W}) = \frac{1}{T} \sum_{t=1}^T l(\mathbf{X}(t), \mathbf{W}) = \frac{1}{T} \sum_{t=1}^T \ln \hat{p}_{\mathbf{X}}(\mathbf{X}(t), \mathbf{W}) \quad (7-29)$$

注意，当只有一个观察向量样本，即  $T = 1$  时，按式 (7-29) 进行  $\mathbf{W}$  学习的算法是随机梯度算法 (见 7.4 节)；若  $T > 1$  则是批处理算法。

应该指出，为计算最大似然目标函数  $L_{\text{ML}}(\mathbf{W})$  或  $\tilde{L}_{\text{ML}}(\mathbf{W})$ ，都需要预先知道各信号源的 pdf  $p_i(S_i)$  (见式 (7-24))。若关于其先验知识不够充分，则必须进行假设或在学习过程中予以确定 详见 7.7 节。

### 7.3.3 统计独立性目标函数

已知  $\mathbf{Y}(t) = \mathbf{W}\mathbf{X}(t)$  (式 7-5) 设  $\mathbf{Y}$  的 pdf 是  $p_{\mathbf{Y}}(\mathbf{Y})$ ， $\mathbf{Y}$  的各个分量的 pdf 为  $p_i(y_i)$ ，且设  $\hat{p}_{\mathbf{Y}}(\mathbf{Y}) = \prod \hat{p}_i(y_i)$  这样可以用  $p_{\mathbf{Y}}(\mathbf{Y})$  和  $\hat{p}_{\mathbf{Y}}(\mathbf{Y})$  之间的 KL 发散度作为  $\mathbf{Y}(t)$  各分量之间统计独立性的度量 (见 7.3.1 小节 (2))，据此构成的目标函数用  $L_{\text{SI}}(\mathbf{W})$  表示 即有

$$L_{\text{SI}}(\mathbf{W}) = KL(p_{\mathbf{Y}}(\mathbf{Y}) \parallel \hat{p}_{\mathbf{Y}}(\mathbf{Y})) = \int_{\mathbf{Y}} p_{\mathbf{Y}}(\mathbf{Y}) \ln \left[ \frac{p_{\mathbf{Y}}(\mathbf{Y})}{\hat{p}_{\mathbf{Y}}(\mathbf{Y})} \right] d\mathbf{Y} \quad (7-30)$$

可以看到，当且仅当  $p_{\mathbf{Y}}(\mathbf{Y}) = \hat{p}_{\mathbf{Y}}(\mathbf{Y}) = \prod \hat{p}_i(y_i)$  时， $L_{\text{SI}}(\mathbf{W})$  等于 0 这时  $\mathbf{Y}(t)$  的各个分量统计独立。如果取此为目标函数 只需求得  $\mathbf{W}$  使  $L_{\text{SI}}(\mathbf{W})$  达到极小值 即求得 ICA 的解。

当只有  $T$  个观察向量  $\mathbf{X}(t)$ ， $t = 1 \sim T$ ， $\mathbf{Y}(t)$  也只有  $T$  个。这时可用近似的目标函数  $L_{\text{SI}}(\mathbf{W})$ 。

$$\tilde{L}_{\text{SI}}(\mathbf{W}) = \frac{1}{T} \sum_{t=1}^T \ln \frac{p_{\mathbf{Y}}(\mathbf{Y}(t))}{\hat{p}_{\mathbf{Y}}(\mathbf{Y}(t))} \quad (7-31)$$

对于平稳随机向量的情况，

$$\lim_{T \rightarrow \infty} \tilde{L}_{\text{SI}}(\mathbf{W}) = L_{\text{SI}}(\mathbf{W}) \quad (7-32)$$

$\tilde{L}_{\text{SI}}(\mathbf{W})$  可以计算如下。由于  $\mathbf{Y}(t) = \mathbf{W}\mathbf{X}(t)$ ，根据式 (7-16) 可得

$$\ln \frac{p_{\mathbf{Y}}(\mathbf{Y}(t))}{\hat{p}_{\mathbf{Y}}(\mathbf{Y}(t))} = -\ln |\det \mathbf{W}| - \sum_{i=1}^N \ln \hat{p}_i(y_i(t)) \Big|_{\mathbf{Y}(t)=\mathbf{W}\mathbf{X}(t)} + \ln p_{\mathbf{X}}(\mathbf{X}(t))$$

此式最右侧的  $\ln p_{\mathbf{X}}(\mathbf{X}(t))$  与  $\mathbf{W}$  无关，因而在目标函数中可以将其作为一个常数，这时式 (7-31) 可以写成下列形式：

$$\tilde{L}_{SI}(\mathbf{W}) = -\ln |\det \mathbf{W}| - \frac{1}{T} \sum_{t=1}^T \sum_{i=1}^N \ln \hat{p}_i(y_i(t)) \Big|_{Y(t)=\mathbf{W}\mathbf{X}(t)} - \tilde{H}(\mathbf{X}) \quad (7-33)$$

其中

$$\tilde{H}(\mathbf{X}) = -\frac{1}{T} \sum_{t=1}^T \ln p_{\mathbf{X}}(\mathbf{X}(t))$$

使  $L_{SI}(\mathbf{W})$  达到极小值的  $\hat{\mathbf{W}}$  即是 ICA 的解。

可以将  $\tilde{L}_{SI}(\mathbf{W})$  与上一小节的最大似然目标函数  $\tilde{L}_{ML}(\mathbf{W})$  作一对比。根据式 (7-24)，式 (7-29) 可以改写成下列形式：

$$\tilde{L}_{ML}(\mathbf{W}) = \ln |\det \mathbf{W}| + \frac{1}{T} \sum_{t=1}^T \sum_{i=1}^N \ln p_i(S_i(t)) \Big|_{S(t)=\mathbf{W}\mathbf{X}(t)} \quad (7-34)$$

对比式 (7-33) 和式 (7-34) 可以看到 如果将各  $y_i$  的 pdf 取得与各源信号  $S_i$  的 pdf 一致 即

$$\hat{p}_i(y_i(t)) = p_i(S_i(t)) \Big|_{S_i(t)=y_i(t)}, \quad i = 1 \sim N \quad (7-35)$$

则这两种目标函数具有下列关系：

$$L_{ML}(\mathbf{W}) = -L_{SI}(\mathbf{W}) - H(\mathbf{X}) \quad (7-36)$$

ICA 的解  $\hat{\mathbf{W}}$  如使  $\tilde{L}_{SI}(\mathbf{W})$  达到极小 即使  $L_{ML}(\mathbf{W})$  达到极大 二者完全等价。在求  $\hat{\mathbf{W}}$  的学习算法中可将与  $\mathbf{W}$  无关的  $\tilde{H}(\mathbf{X})$  略去不顾并用式 (7-33) 来计算目标函数，这种处理方法更加方便。

### 7.3.4 信息最大化(最大熵)目标函数

按文献[2]的思路 求解 ICA 问题时，对于每个观察向量  $\mathbf{X} = [x_1, x_2, \dots, x_N]^T$  先通过线性变换求一个中间向量  $\mathbf{Z} = [z_1, z_2, \dots, z_N]^T = \mathbf{W}\mathbf{X}$ 。然后通过非线性变换  $y_i = g_i(z_i)$  求得输出向量  $\mathbf{Y} = [y_1, y_2, \dots, y_N]^T$ 。再针对  $\mathbf{Y}$  建立一个目标函数，通过学习求得一最佳  $\mathbf{W}$  使此目标函数达到极值，该  $\mathbf{W}$  即是 ICA 的解。此思路的初衷是模仿单层前向神经网络  $\mathbf{X}$  和  $\mathbf{Y}$  分别作为网络的输入和输出。关键在于如何选择目标函数和各个非线性函数  $g_i(\cdot)$ 。该文献选择  $\mathbf{Y}$  的熵作为目标函数 表示为  $L_H(\mathbf{W})$ ，即有

$$L_H(\mathbf{W}) = H(\mathbf{Y}) = -E[\ln p_Y(\mathbf{Y})] \quad (7-37)$$

其中  $p_Y(\mathbf{Y})$  是  $\mathbf{Y}$  的 pdf。当只有  $T$  个观察向量  $\mathbf{X}(t), t = 1 \sim T$  时， $\mathbf{Y}(t)$  也只有  $T$  个。这时可用下列近似目标函数  $L_H(\mathbf{W})$  替代  $L_H(\mathbf{W})$

$$\tilde{L}_H(\mathbf{W}) = -\frac{1}{T} \sum_{t=1}^T \ln p_Y(\mathbf{Y}(t)) \quad (7-38)$$

且  $\lim_{T \rightarrow \infty} \tilde{L}_H(\mathbf{W}) = L_H(\mathbf{W})$ 。选择熵作为目标函数是因为熵是一个随机量无序性的度量，如果  $\mathbf{Y}$  的各分量统计独立性越高则相应  $\mathbf{Y}$  的熵越大 所以只需求得使  $\tilde{L}_H(\mathbf{W})$  达到最大的  $\hat{\mathbf{W}}$  即求得了 ICA 的解。

参照 7.3.1 小节 4 式 (7-38) 中的  $p_Y(\mathbf{Y}(t))$  可计算如下。如用  $y_i$  表示  $dg_i(z_i)/dz_i = dy_i/dz_i$  则式 (7-21) 的雅可比阵  $\mathbf{J}(\mathbf{G})$  的元素  $J_{ij}$  可表示为

$$J_{ij} = y'_i w_{ij}, \quad i = 1 \sim N, j = 1 \sim N \quad (7-39)$$

易于证明  $\mathbf{J}(\mathbf{G})$  的行列式可用下式计算：

$$\det \mathbf{J}(\mathbf{G}) = \det \mathbf{W} \prod_{i=1}^N y'_i \quad (7-40)$$

这样,按照式(7-20), $p_Y(\mathbf{Y}(t))$ 可表示为

$$p_Y(\mathbf{Y}(t)) = \left| \det \mathbf{W} \prod_{i=1}^N y'_i(t) \right|^{-1} \cdot p_X(\mathbf{X}(t)) \quad (7-41)$$

为简洁起见  $\mathbf{Y}(t) = \mathbf{G}(\mathbf{X}(t))$  这一条件在此式中不再标明。将式(7-41)代入式(7-38)得到

$$L_H(\mathbf{W}) = \ln |\det \mathbf{W}| + \frac{1}{T} \sum_{t=1}^T \sum_{i=1}^N \ln y'_i(t) + \frac{-1}{T} \sum_{t=1}^T \sum_{i=1}^N \ln p_{X_i}(\mathbf{X}(t)) \quad (7-42)$$

注意此式最右侧项等于  $\tilde{H}(\mathbf{X})$ 。式(7-33)和式(7-36)对比后可以看到,如条件

$$y'_i(t) = \hat{p}_i(y_i(t)) = p_i(S_i(t)) \Big|_{S_i(t)=y_i(t)} \quad i = 1 \sim N \quad (7-43)$$

得到满足 则

$$L_H(\mathbf{W}) = -L_{SI}(\mathbf{W}) = L_{ML}(\mathbf{W}) + H(\mathbf{X}) \quad (7-44)$$

如果略去与  $\mathbf{W}$  无关的  $\tilde{H}(\mathbf{X})$ , 上式中的三种目标函数是一致的,ICA 的解  $\hat{\mathbf{W}}$  应使  $\tilde{L}_H(\mathbf{W})$  和  $L_{ML}(\mathbf{W})$  达到极大值,使  $\tilde{L}_{SI}(\mathbf{W})$  达到极小值。

A. J. Bell 等在 1995 年提出  $\tilde{L}_H(\mathbf{W})$  时模仿神经网络的办法,将非线性函数  $y_i = g_i(z_i)$  选择为 Sigmoid 函数<sup>[2]</sup> 即有

$$y_i = g(z_i) = \frac{1}{1 + e^{-z_i}} \quad \text{或} \quad y_i = g(z_i) = \tanh(z_i), \quad \forall i$$

这时  $y_i$  的取值范围为(0,1)或(-1,1)。相应地

$$y'_i = \frac{dg(z_i)}{dz_i} = \hat{p}_i(y_i) = y_i(1 - y_i) \quad \text{或} \quad y'_i = \hat{p}_i(y_i) = 1 - y_i^2$$

这种类型的 pdf 具有超高斯特性,所以文献[2]用它来解决同样具有超高斯特性 pdf 的语音和音乐信号 BSS 问题时效果很好。相反,用它来解决具有次高斯特性 pdf 的两个均匀分布随机信号 BSS 问题时不能奏效。这样,解决 BSS 问题仍归结于对被分离信号源 pdf  $p_i(S_i)$ (或等效的  $\hat{p}_i(y_i)$ ) 的确定或在学习过程中对它的确定。很多实验表明,关键在于确定这些 pdf 是超高斯的或次高斯的,而其具体形式细节对分离效果的影响往往不大。

文献[22]对于各种目标函数进行了列举和对比,可以参考。其他目标函数将在 7.4 节讨论。

由于各种目标函数的一致性且由于  $\tilde{H}(\mathbf{X})$  项与  $\mathbf{W}$  无关 所以为简洁计 后续章节中用下列目标函数  $\tilde{L}(\mathbf{W})$  来计算(参考式(7-33)、(7-34)、(7-44)等):

$$\tilde{L}(\mathbf{W}) = -\ln |\det \mathbf{W}| - \frac{1}{T} \sum_{t=1}^T \sum_{i=1}^N \ln \hat{p}_i(y_i(t)) \Big|_{\mathbf{Y}(t)=\mathbf{W}\mathbf{X}(t)} \quad (7-45)$$

如  $\hat{\mathbf{W}}$  使  $\tilde{L}(\mathbf{W})$  达到极小,  $\hat{\mathbf{W}}$  即是 ICA 的解。

## 7.4 ICA 的学习算法

建立目标函数  $\tilde{L}(\mathbf{W})$  后 即需一种学习算法 通过迭代计算求得使  $\tilde{L}(\mathbf{W})$  (式(7-45)) 达到极小值的解。一种好的学习算法应保证所求得解正是所需的 BSS 问题的解。这种学

习算法还应该是高效的——计算简单且收敛速度快。通过上世纪最后 10 年的大量研究工作，包括理论分析和模拟实验，BSP 研究学者普遍认为相对梯度（relative gradient）学习算法可以达到上面提出的目标。关于此算法应说明以下几点。

(1) 关于这一算法还有其他各种名称和解释，例如文献 [23] 从微分几何的角度导出这一算法并称之为自然梯度（natural gradient）算法（见 7.4.3 小节），文献 [17] 也从类似的角度予以解释（见 7.4.4 小节）。

(2) 实践结果表明这一算法效果很好。

(3) 这种算法的起源来自对于人工神经网络的模拟<sup>[15,2]</sup>，只是在模拟实验中取得很好效果以后，研究者才试图对其作出理论解释，其实这仍是一种启发式的算法。其他有关的启发式学习算法将在 7.4.5 小节讨论。

(4) 这种算法还有一种优良的特性，即等价变化性（equivariant）这在 7.4.2 小节讨论。

(5) 这种算法的效果与学习步幅的自适应有关。7.4.6 小节将讨论步幅的自适应学习问题。

### 7.4.1 相对梯度学习算法

在一般人工神经网络中，所有权参数构成一个向量，其学习算法中利用目标函数对于权参数向量的梯度。在 ICA 中  $\mathbf{W}$  是一个  $N \times N$  维矩阵，可以将其改写为一个  $N^2 \times 1$  维列向量来进行处理，后述 7.4.5 小节和 7.4.6 小节中关于 ICA 的稳定性和精度分析中即取此方法。而在用迭代算法求使  $\tilde{L}(\mathbf{W})$  达极小值的最优解时，仍将  $\mathbf{W}$  作为一个矩阵来处理。下面先给出若干定义。

设有一个  $N \times N$  维矩阵  $\Phi = (\varphi_{ij})$  其迹  $\text{tr} \Phi$  为

$$\text{tr} \Phi = \sum_{i=1}^N \varphi_{ii} \quad (7-46)$$

设有两个  $N \times N$  维矩阵  $\mathbf{A} = (a_{ij})$  和  $\mathbf{B} = (b_{ij})$  其内积用  $\langle \mathbf{A} | \mathbf{B} \rangle$  表示且定义如下：

$$\langle \mathbf{A} | \mathbf{B} \rangle = \text{tr}(\mathbf{A}\mathbf{B}^T) \quad (7-47)$$

对于一个目标函数  $\tilde{L}(\mathbf{W})$ ，它相对于  $\mathbf{W}$  的梯度也是一个  $N \times N$  维矩阵，用  $\nabla_{\mathbf{W}} \tilde{L}(\mathbf{W})$  表示，若  $\mathbf{W} = (w_{ij})$ ，则  $\nabla_{\mathbf{W}} \tilde{L}(\mathbf{W}) = (\partial \tilde{L}(\mathbf{W}) / \partial w_{ij})$ 。这样，采用此梯度可以求得  $\mathbf{W}$  变化为  $\mathbf{W} + \Delta \mathbf{W}$  时，其中  $\Delta \mathbf{W} = (\Delta w_{ij})$  表示一“微小变化”矩阵）

$$\tilde{L}(\mathbf{W} + \Delta \mathbf{W}) = \tilde{L}(\mathbf{W}) + \sum_{i=1}^N \sum_{j=1}^N \frac{\partial \tilde{L}(\mathbf{W})}{\partial w_{ij}} \Delta w_{ij} + O(\Delta \mathbf{W}) \quad (7-48a)$$

$$\tilde{L}(\mathbf{W} + \Delta \mathbf{W}) \approx \tilde{L}(\mathbf{W}) + \langle \nabla_{\mathbf{W}} \tilde{L}(\mathbf{W}) | \Delta \mathbf{W} \rangle \quad (7-48b)$$

其中  $O(\Delta \mathbf{W})$  表示较  $\|\Delta \mathbf{W}\|$  小一个量级的微小量（ $\|\Delta \mathbf{W}\| = \langle \Delta \mathbf{W} | \Delta \mathbf{W} \rangle^{\frac{1}{2}}$ ）在近似表式中可以被略去。按照常规的最陡下降梯度算法，为了使每一步迭代节拍  $k$ ， $\mathbf{W}(k)$  变至  $\mathbf{W}(k) + \Delta \mathbf{W}(k)$  时， $\tilde{L}(\mathbf{W}(k) + \Delta \mathbf{W}(k))$  皆小于  $\tilde{L}(\mathbf{W}(k))$  应该令

$$\Delta \mathbf{W}(k) = -\alpha(k) \nabla_{\mathbf{W}} \tilde{L}(\mathbf{W}) \Big|_{\mathbf{W}=\mathbf{W}(k)} \quad (7-49)$$

其中  $0 < \alpha(k) \ll 1$ ，是随  $k$  而变化的步幅。

下面讨论相对梯度。设  $\mathbf{I}$  是  $N \times N$  维单位阵， $\boldsymbol{\varepsilon} = (\varepsilon_{ij})$  是一个  $N \times N$  维的微小矩阵（即



满足  $|\varepsilon_{ij}| \ll 1, \forall i, j$ 。这样当  $\mathbf{W}$  变为  $(\mathbf{I} + \boldsymbol{\varepsilon})\mathbf{W}$  时,  $\tilde{L}(\mathbf{W})$  的变化可用下式计算:

$$\tilde{L}((\mathbf{I} + \boldsymbol{\varepsilon})\mathbf{W}) = \tilde{L}(\mathbf{W}) + \langle \nabla_{\boldsymbol{\varepsilon}} \tilde{L}(\mathbf{W}) | \boldsymbol{\varepsilon} \rangle + O(\boldsymbol{\varepsilon}) \quad (7-50)$$

其中  $O(\boldsymbol{\varepsilon})$  是较  $\|\boldsymbol{\varepsilon}\|$  小一个数量级的微小量,  $\nabla_{\boldsymbol{\varepsilon}} \tilde{L}(\mathbf{W})$  称为相对梯度, 其定义如下:

$$\nabla_{\boldsymbol{\varepsilon}} \tilde{L}(\mathbf{W}) = \frac{\partial \tilde{L}((\mathbf{I} + \boldsymbol{\varepsilon})\mathbf{W})}{\partial \varepsilon_{ij}} \quad (7-51)$$

这样

$$L((\mathbf{I} + \boldsymbol{\varepsilon})\mathbf{W}) \approx L(\mathbf{W}) + \langle \nabla_{\boldsymbol{\varepsilon}} L(\mathbf{W}) | \boldsymbol{\varepsilon} \rangle \quad (7-52)$$

如果设  $\boldsymbol{\varepsilon}\mathbf{W} = \Delta\mathbf{W}$  则由式 (7-48b) 得到

$$L((\mathbf{I} + \boldsymbol{\varepsilon})\mathbf{W}) \approx L(\mathbf{W}) + \langle \nabla_{\mathbf{W}} L(\mathbf{W}) | \boldsymbol{\varepsilon}\mathbf{W} \rangle \quad (7-53)$$

对比式 (7-52) 和式 (7-53) 可知  $\langle \nabla_{\boldsymbol{\varepsilon}} \tilde{L}(\mathbf{W}) | \boldsymbol{\varepsilon} \rangle = \langle \nabla_{\mathbf{W}} \tilde{L}(\mathbf{W}) | \boldsymbol{\varepsilon}\mathbf{W} \rangle$  引用式 (7-47) 可得

$$\text{tr}[\nabla_{\mathbf{W}} \tilde{L}(\mathbf{W}) (\boldsymbol{\varepsilon}\mathbf{W})^T] = \text{tr}[\nabla_{\boldsymbol{\varepsilon}} \tilde{L}(\mathbf{W}) \boldsymbol{\varepsilon}^T]$$

将上式左右对照, 即得到相对梯度与普通梯度之间关系如下:

$$\nabla_{\boldsymbol{\varepsilon}} \tilde{L}(\mathbf{W}) = \nabla_{\mathbf{W}} \tilde{L}(\mathbf{W}) \mathbf{W}^T \quad (7-54)$$

在按照相对梯度进行迭代计算时, 每一节拍  $k$  的  $\mathbf{W}(k)$  将变至  $(\mathbf{I} + \boldsymbol{\varepsilon}(k))\mathbf{W}(k)$  其中

$$\boldsymbol{\varepsilon}(k) = -\alpha(k) \nabla_{\boldsymbol{\varepsilon}} \tilde{L}(\mathbf{W}) \Big|_{\mathbf{W}=\mathbf{W}(k)} \quad (7-55)$$

注意到  $\Delta\mathbf{W}(k) = \boldsymbol{\varepsilon}(k)\mathbf{W}(k)$  再引用式 (7-54) 即得

$$\Delta\mathbf{W}(k) = -\alpha(k) \nabla_{\mathbf{W}} \tilde{L}(\mathbf{W}) \mathbf{W}^T \mathbf{W} \Big|_{\mathbf{W}=\mathbf{W}(k)} \quad (7-56)$$

这就是按相对梯度法求得的  $\mathbf{W}$  迭代计算公式。

现在, 将式 (7-45) 代入式 (7-56) 首先得到

$$\nabla_{\mathbf{W}} \tilde{L}(\mathbf{W}) = -\nabla_{\mathbf{W}} \ln |\det \mathbf{W}| + \nabla_{\mathbf{W}} \sum_{i=1}^N \left( \frac{-1}{T} \sum_{t=1}^T \ln \hat{p}_i(y_i(t)) \right) \Big|_{\mathbf{Y}(t) = \mathbf{W}\mathbf{X}(t)}$$

上式右侧第一项可用下式计算 (推导从略):

$$\nabla_{\mathbf{W}} \ln |\det \mathbf{W}| = \mathbf{W}^{-T}$$

其中  $\mathbf{W}^{-T} = (\mathbf{W}^{-1})^T$ 。注意到  $y_i(t) = \sum_{j=1}^N w_{ij} x_j(t)$ , 且设

$$\phi = \sum_{i=1}^N \left( \frac{-1}{T} \sum_{t=1}^T \ln \hat{p}_i(y_i(t)) \right) \Big|_{\mathbf{Y}(t) = \mathbf{W}\mathbf{X}(t)}$$

则前式右侧第二项等于一个  $N \times N$  维矩阵  $\left( \frac{\partial \phi}{\partial w_{ij}} \right)$ , 注意到

$$\frac{\partial \ln \hat{p}_i(y_i(t))}{\partial w_{ij}} = \frac{\hat{p}'_i(y_i(t))}{\hat{p}_i(y_i(t))} x_j(t)$$

其中  $\hat{p}'_i(y_i(t))$  是  $\hat{p}_i(y_i(t))$  对  $y_i(t)$  的导数。如果假设

$$\boldsymbol{\varphi}(\mathbf{Y}(t)) = \left[ -\frac{\hat{p}'_1(y_1(t))}{\hat{p}_1(y_1(t))}, -\frac{\hat{p}'_2(y_2(t))}{\hat{p}_2(y_2(t))}, \dots, -\frac{\hat{p}'_N(y_N(t))}{\hat{p}_N(y_N(t))} \right]^T \quad (7-57)$$

已知

$$\mathbf{X}(t) = [x_1(t), x_2(t), \dots, x_N(t)]^T$$

经过简单推导, 即得到

$$\left(\frac{\partial \phi}{\partial w_{ij}}\right) = \frac{1}{T} \sum_{t=1}^T \boldsymbol{\varphi}(\mathbf{Y}(t)) \mathbf{X}^T(t)$$

这样，式 (7-56) 可以写成

$$\Delta \mathbf{W}(k) = \alpha(k) \left[ \mathbf{W}^{-T}(k) - \frac{1}{T} \sum_{t=1}^T \boldsymbol{\varphi}(\mathbf{Y}(t)) \mathbf{X}^T(t) \right] \mathbf{W}^T(k) \mathbf{W}(k) \quad (7-58)$$

注意到  $\mathbf{W}^{-T}(k) \mathbf{W}^T(k) = \mathbf{I}$  且因为  $\mathbf{Y}(t) = \mathbf{W}(k) \mathbf{X}(t)$  所以  $\mathbf{X}^T(t) \mathbf{W}^T(k) = \mathbf{Y}^T(t)$  这样上式可以写成下列非常简洁的形式：

$$\Delta \mathbf{W}(k) = \alpha(k) \left[ \mathbf{I} - \frac{1}{T} \sum_{t=1}^T \boldsymbol{\varphi}(\mathbf{Y}(t)) \mathbf{Y}^T(t) \right] \mathbf{W}(k) \quad (7-59)$$

这是 ICA 学习算法中最重要的一个关键公式。和其他神经网络的学习算法可以分成离线批处理和在线自适应两种方式一样，ICA 学习算法也可按这两种方法实施。如果在学习前预先搜集到  $T$  个观察向量  $\mathbf{X}(t), t = 1 \sim T$  再从随机初值  $\mathbf{W}(0)$  出发，按式 (7-59) 以节拍  $k$  进行迭代计算，来求最佳解  $\mathbf{W}$  即构成离线批处理算法。如果每得到一个观察向量  $\mathbf{X}(t)$ ，就进行一次学习，这时样本数  $T = 1$  且迭代计算节拍  $k$  与  $\mathbf{X}(t)$  的时序  $t$  相重合，如果将二者皆用  $k$  表示，则可以得到如下 ICA 的在线自适应学习算法：

$$\Delta \mathbf{W}(k) = \alpha(k) [\mathbf{I} - \boldsymbol{\varphi}(\mathbf{Y}(k)) \mathbf{Y}^T(k)] \mathbf{W}(k) \quad (7-60)$$

和神经网络情况相同，这是一种随机梯度算法，所以对  $\alpha(k)$  的选择尤需注意。事实上对于预先搜集到  $T$  个样本的离线情况，也可用式 (7-60) 进行学习，只需按时序  $k$  从训练集中每次取出一个观察样本  $\mathbf{X}(k)$  即可。这时施行的是离线随机梯度算法。此外，应注意在式 (7-59) 的批处理算法中，样本时序  $t$  和迭代时序  $k$  是不一致的，在该式中， $\mathbf{Y}(t) = \mathbf{W}(k) \mathbf{X}(t)$  的表达不够确切，更确切的表示应该是

$$\mathbf{Y}(t, k) = \mathbf{W}(k) \mathbf{X}(t) \quad t = 1 \sim T, k = 0, 1, \dots \quad (7-61)$$

这时离线批处理 ICA 学习算法可表示为

$$\left. \begin{aligned} \mathbf{W}(k+1) &= \mathbf{W}(k) + \Delta \mathbf{W}(k), \quad k = 0, 1, 2, \dots \\ \Delta \mathbf{W}(k) &= \alpha(k) \left[ \mathbf{I} - \frac{1}{T} \sum_{t=1}^T \boldsymbol{\varphi}(\mathbf{Y}(t, k)) \mathbf{Y}^T(t, k) \right] \mathbf{W}(k) \end{aligned} \right\} \quad (7-62)$$

其中  $\mathbf{W}(0)$  为随机初值  $\alpha(k)$  为经过选择的步幅  $\mathbf{Y}(t, k)$  按式 (7-61) 计算。

#### 7.4.2 相对梯度学习算法的等价变化性

ICA 的根本目标是实现 BSS，即求得各个信号源，而不在于求得反混合阵本身。已知通过迭代计算使  $\mathbf{W}(k)$  收敛于最佳解时，通过变换  $\mathbf{Y} = \mathbf{W}\mathbf{X}$  将使  $\mathbf{Y}$  近似于源信号向量  $\mathbf{S}$ 。这就是说通过 ICA 算法的迭代计算，将使  $\mathbf{Y}(t, k)$  随着  $k$  的增加逼近  $\mathbf{S}(t)$ 。这一过程可展示如下。已知  $\mathbf{X}(t) = \mathbf{A}\mathbf{S}(t)$ ， $\mathbf{Y}(t, k) = \mathbf{W}(k) \mathbf{X}(t)$  因此  $\mathbf{Y}(t, k) = \mathbf{W}(k) \mathbf{A} \mathbf{S}(t)$ 。设

$$\mathbf{C}(k) = \mathbf{W}(k) \mathbf{A} \quad (7-63)$$

$\mathbf{C}(k)$  是一个  $N \times N$  维矩阵，使得

$$\mathbf{Y}(t, k) = \mathbf{C}(k) \mathbf{S}(t) \quad (7-64)$$

对式 (7-62) 中两个方程式左右侧皆右乘  $\mathbf{A}$ ，再将其中的第二式代入第一式，即得到  $\mathbf{C}(k)$  满足的迭代计算方程，即有

$$\mathbf{C}(k+1) = \mathbf{C}(k) + \alpha(k) \left[ \mathbf{I} - \frac{1}{T} \sum_{t=1}^T \boldsymbol{\varphi}(\mathbf{Y}(t,k)) \mathbf{Y}^T(t,k) \right] \mathbf{C}(k) \quad (7-65)$$

其初始条件是

$$\mathbf{C}(0) = \mathbf{W}(0)\mathbf{A} \quad (7-66)$$

可以看到，表面上 ICA 做的迭代计算（式 7-61 和式 7-62）是从随机初值  $\mathbf{W}(0)$  出发，按节拍  $k = 0, 1, 2, \dots$  由  $\mathbf{W}(k)$  及  $\mathbf{X}(t)$  求  $\mathbf{Y}(t,k)$  再由  $\mathbf{Y}(t,k)$  及  $\mathbf{W}(k)$  求  $\mathbf{W}(k+1)$  依次随  $k$  循环。而此迭代的实质（式 7-64）~ 式(7-66)）是：从随机初值  $\mathbf{C}(0)$  出发，按节拍  $k = 0, 1, 2, \dots$  由  $\mathbf{C}(k)$  及  $\mathbf{S}(t)$  求  $\mathbf{Y}(t,k)$  再由  $\mathbf{Y}(t,k)$  及  $\mathbf{C}(k)$  求  $\mathbf{C}(k+1)$  依次循环。由此过程可以看到一个非常有用而且有趣的特点： $\mathbf{A}$  的作用只是决定迭代的初值  $\mathbf{C}(0)$ ，其作用与  $\mathbf{W}(0)$  完全一样，而在以后的过程中迭代的性能只取决于  $\boldsymbol{\varphi}(\mathbf{Y}(t,k))$ ，它取决于源信号的 pdf 或者对于此 pdf 的一种假设。这就是说，为了实现 BSS 混合阵  $\mathbf{A}$  的作用是很弱的；起关键性作用的是源信号的统计特性以及我们能否把握它或在学习过程中决定它。上述特性即称为等价变换性。

对于在线自适应随机梯度算法的情况，迭代节拍  $k$  和观察向量时序  $t$  统一用  $k$  表示。根据式 7-60），迭代计算公式可表示为：按照时序  $k = 0, 1, 2, \dots$  依次得到观察向量  $\mathbf{X}(k)$  以及随机设置的初值  $\mathbf{W}(0)$ ，然后按下式进行迭代计算：

$$\left. \begin{aligned} \mathbf{C}(0) &= \mathbf{W}(0)\mathbf{A} \\ \mathbf{Y}(k) &= \mathbf{W}(k)\mathbf{X}(k) = \mathbf{W}(k)\mathbf{A}\mathbf{S}(k) = \mathbf{C}(k)\mathbf{S}(k) \\ \mathbf{C}(k+1) &= \mathbf{C}(k) + \alpha(k) [\mathbf{I} - \boldsymbol{\varphi}(\mathbf{Y}(k))\mathbf{Y}^T(k)] \mathbf{C}(k) \end{aligned} \right\} \quad (7-67)$$

这就是在线自适应 ICA 算法的等价变化特性。注意，由于  $\mathbf{A}$  为未知，所以  $\mathbf{C}(0)$  为未知，式(7-67) 的计算是“隐含”的。实际执行的计算是按照式 7-60) 进行的。

### 7.4.3 自然梯度学习算法

迄至本节为止，我们讨论的参数空间都是欧氏空间。无论参数  $\mathbf{W}$  是向量还是矩阵（下面只讨论  $\mathbf{W}$  为  $N \times N$  维矩阵的情况），以下两个特点对于  $\mathbf{W}$  为欧氏空间参数时神经网络的学习算法构成至关重要。第一 设  $\Delta\mathbf{W} = (\Delta w_{ij})$  是  $\mathbf{W}$  附近的一个微小变化  $N \times N$  维矩阵 则  $\mathbf{W}$  的模平方可以用其内积求得（参见式 7-47)）即

$$\|\Delta\mathbf{W}\|^2 = \langle \Delta\mathbf{W} | \Delta\mathbf{W} \rangle = \text{tr}[\Delta\mathbf{W}\Delta\mathbf{W}^T] = \sum_{i=1}^N \sum_{j=1}^N (\Delta w_{ij})^2 \quad (7-68)$$

注意，在欧氏空间中， $\|\mathbf{W}\|^2$  之值对任一  $\mathbf{W}$  都不改变。第二 设有一个以  $\mathbf{w}$  为变元的目标函数  $\tilde{L}(\mathbf{W})$  则对于任何  $\mathbf{W}$ ， $\tilde{L}(\mathbf{W})$  的最陡下降方向是  $-\nabla_{\mathbf{W}}\tilde{L}(\mathbf{W}) = (-\partial L(\mathbf{W})/\partial w_{ij})$ 。应强调，在欧氏空间中 最陡下降方向不随  $\mathbf{w}$  改变。这是神经网络学习算法的一个基础。

文献[23] 指出，为解决 BSS 问题而提出的 ICA 算法中 参数元  $\mathbf{W}$  的空间是一个非欧的黎曼 (Riemann) 空间。在此空间中，上列两个特点不复存在，而且此空间中所有  $N \times N$  维非奇矩阵构成一个李 Lie 群 其运算符合李代数规则。下面略述其要点。

(1) 在黎曼空间中， $\Delta\mathbf{W}$  的模平方  $\|\Delta\mathbf{W}\|^2$  随所取的位置点  $\mathbf{W}$  而改变，所以必须标明其位置  $\mathbf{W}$  即有

$$\|\Delta \mathbf{W}\|_{\mathbf{W}}^2 = \langle \Delta \mathbf{W} | \Delta \mathbf{W} \rangle_{\mathbf{W}} = \sum_{i,j=1}^N \sum_{k,l=1}^N \Delta w_{ij} \Delta w_{kl} \zeta_{ij,kl}(\mathbf{W}) \quad (7-69)$$

其中  $\zeta_{ij,kl}(\mathbf{W})$  为  $\mathbf{W}$  的函数。如果

$$\zeta_{ij,kl}(\mathbf{W}) = \delta_{ik} \delta_{jl} \quad \forall \mathbf{W}$$

成立, 则式 (7-69) 退化为式 (7-68), 黎曼空间退化为欧氏空间。

(2) 在黎曼空间中有一个特殊点  $\mathbf{W} = \mathbf{I}$ , 在此点附近表现出欧氏空间的特性, 即有

$$\|\Delta \mathbf{W}\|_{\mathbf{W}=\mathbf{I}}^2 = \langle \Delta \mathbf{W} | \Delta \mathbf{W} \rangle \Big|_{\mathbf{W}=\mathbf{I}} = \sum_{i=1}^N \sum_{j=1}^N (\Delta w_{ij})^2 \quad (7-70)$$

其次, 在  $\mathbf{W} = \mathbf{I}$  附近, 任何目标函数  $\tilde{L}(\mathbf{W})$  相对于  $\mathbf{W}$  的梯度取负号后即是最陡下降方向。

若  $\mathbf{W} \neq \mathbf{I}$  则不能由  $\tilde{L}(\mathbf{W})$  对  $\mathbf{W}$  的梯度决定最陡下降方向。

(3) 设  $\Phi$  是一个任意  $N \times N$  维矩阵, 则在黎曼空间中

$$\langle \Delta \mathbf{W} | \Delta \mathbf{W} \rangle_{\mathbf{W}} = \langle \Delta \mathbf{W} \Phi | \Delta \mathbf{W} \Phi \rangle_{\mathbf{W} \Phi} \quad (7-71)$$

成立<sup>[24]</sup> 这称为李群不变性。

现在利用这些知识求 ICA 算法在黎曼空间中的解。问题同样归结为用迭代算法求  $\mathbf{W}$ , 使得目标函数  $\tilde{L}(\mathbf{W})$  达到极小值。但它与在欧氏空间中求解不一样, 需要在迭代计算的每一步在微小变化矩阵  $\Delta \mathbf{W}$  的模平方值  $\|\Delta \mathbf{W}\|_{\mathbf{W}}^2$  保持一定的条件下, 求  $\Delta \mathbf{W}$  使得  $\tilde{L}(\mathbf{W} + \Delta \mathbf{W}) - \tilde{L}(\mathbf{W})$  达到最小。为此需求得  $\tilde{L}(\mathbf{W})$  的最陡下降方向。但是, 在黎曼空间中, 若  $\mathbf{W} \neq \mathbf{I}$  则一般的梯度  $\nabla_{\mathbf{W}} \tilde{L}(\mathbf{W}) = (\partial \tilde{L} / \partial w_{ij})_{\mathbf{W} \neq \mathbf{I}}$  取负号后并非最陡下降方向。为此采取下列措施: 令  $\Delta \mathbf{W} = \boldsymbol{\varepsilon} \mathbf{W}$ ,  $\boldsymbol{\varepsilon} = (\varepsilon_{ij})$  是相对微小变化矩阵。这时  $L(\mathbf{W} + \Delta \mathbf{W}) = L((\mathbf{I} + \boldsymbol{\varepsilon})\mathbf{W})$ 。根据李群不变性,

$$\|\Delta \mathbf{W}\|_{\mathbf{W}}^2 = \langle \Delta \mathbf{W} | \Delta \mathbf{W} \rangle_{\mathbf{W}} = \langle \Delta \mathbf{W} \mathbf{W}^{-1} | \Delta \mathbf{W} \mathbf{W}^{-1} \rangle_{\mathbf{W} \mathbf{W}^{-1}} = \langle \boldsymbol{\varepsilon} | \boldsymbol{\varepsilon} \rangle_{\mathbf{I}} = \|\boldsymbol{\varepsilon}\|_{\mathbf{I}}^2$$

这样问题转化为在  $\|\boldsymbol{\varepsilon}\|$  保持不变的条件下求  $\boldsymbol{\varepsilon}$  使  $\tilde{L}((\mathbf{I} + \boldsymbol{\varepsilon})\mathbf{W}) - \tilde{L}(\mathbf{W})$  达到最小。由于  $\boldsymbol{\varepsilon}$  是  $\mathbf{I}$  附近的微小变化阵, 所以  $\tilde{L}((\mathbf{I} + \boldsymbol{\varepsilon})\mathbf{W})$  相对于  $\boldsymbol{\varepsilon}$  的梯度取负号即是最陡下降方向。如记此梯度为  $\nabla_{\boldsymbol{\varepsilon}} \tilde{L}(\mathbf{W})$  则可表示为

$$\nabla_{\boldsymbol{\varepsilon}} \tilde{L}(\mathbf{W}) = \left( \frac{\partial \tilde{L}((\mathbf{I} + \boldsymbol{\varepsilon})\mathbf{W})}{\partial \varepsilon_{ij}} \right) \quad (7-72)$$

这样, 可以求得在任意  $\mathbf{W}$  下的最佳相对微小变化矩阵  $\boldsymbol{\varepsilon}$  为

$$\boldsymbol{\varepsilon} = -\alpha \nabla_{\boldsymbol{\varepsilon}} \tilde{L}(\mathbf{W}) \quad (7-73)$$

相应  $\mathbf{W}$  的变化  $\Delta \mathbf{W}$  可计算如下。由  $\Delta \mathbf{W} = \boldsymbol{\varepsilon} \mathbf{W}$  可得

$$\Delta w_{kl} = \sum_{m=1}^N \varepsilon_{km} w_{ml}, \quad k, l = 1 \sim N \quad (7-74)$$

利用全微分式得到

$$\frac{\partial \tilde{L}((\mathbf{I} + \boldsymbol{\varepsilon})\mathbf{W})}{\partial \varepsilon_{ij}} = \sum_{k,l=1}^N \frac{\partial \tilde{L}((\mathbf{I} + \boldsymbol{\varepsilon})\mathbf{W})}{\partial w_{kl}} \frac{\partial w_{kl}}{\partial \varepsilon_{ij}} \quad (7-75)$$

由式 (7-74) 可知, 当  $k \neq i$  时,  $w_{kl}$  与  $\varepsilon_{ij}$  无关, 因此上式右侧只包含  $k = i$  的项, 可以写成

$$\sum_{l=1}^N \frac{\partial \tilde{L}((\mathbf{I} + \boldsymbol{\varepsilon})\mathbf{W})}{\partial w_{il}} \frac{\partial w_{il}}{\partial \varepsilon_{ij}}$$

再次用式 (7-74), 可得 (令式中  $k = i$ )

$$\frac{\partial w_{il}}{\partial \varepsilon_{ij}} = w_{jl}$$

因此，式 7-75 可表示为

$$\frac{\partial \tilde{L}((\mathbf{I} + \boldsymbol{\varepsilon})\mathbf{W})}{\partial \varepsilon_{ij}} = \sum_{l=1}^N \frac{\partial \tilde{L}(\mathbf{W} + \Delta \mathbf{W})}{\partial w_{il}} w_{jl} \quad (7-76)$$

用  $\partial \tilde{L}(\mathbf{W}) / \partial w_{il}$  表示  $\partial L(\mathbf{W} + \Delta \mathbf{W}) / \partial w_{il}$ ，且如 5.4.1 小节用  $\nabla_{\mathbf{w}} \tilde{L}(\mathbf{W})$  表示矩阵  $(\partial \tilde{L}(\mathbf{W}) / \partial w_{ij})$ ，注意  $[\partial \tilde{L}(\mathbf{W}) / \partial w_{i1}, \partial \tilde{L}(\mathbf{W}) / \partial w_{i2}, \dots, \partial \tilde{L}(\mathbf{W}) / \partial w_{iN}]$  是此矩阵的第  $i$  行向量， $[w_{j1}, w_{j2}, \dots, w_{jN}]$  是矩阵  $\mathbf{W}$  的第  $j$  行向量。引用式 (7-72) 则式 (7-76) 可表示为

$$\nabla_{\boldsymbol{\varepsilon}} \tilde{L}(\mathbf{W}) = \nabla_{\mathbf{w}} \tilde{L}(\mathbf{W}) \mathbf{W}^T \quad (7-77)$$

将此式代入式 (7-73) 且由  $\mathbf{W} = \boldsymbol{\varepsilon} \mathbf{W}$  即得

$$\Delta \mathbf{W} = -\alpha \nabla_{\mathbf{w}} \tilde{L}(\mathbf{W}) \mathbf{W}^T \quad (7-78)$$

这即是自然梯度迭代计算公式<sup>[23,8]</sup>。与式 (7-56) 对比可以看到，自然梯度与相对梯度完全一致，由后者导出的运算公式完全适用于前者。

#### 7.4.4 通过求目标函数梯度的 0 值点来得到 ICA 解的学习算法<sup>[17]</sup>

按照 7.3 节式 (7-45) 给出的目标函数  $\tilde{L}(\mathbf{W})$  如果  $\mathbf{W}$  是微小参数变化矩阵，则可以求得  $\mathbf{W}$  变至  $\mathbf{W} + \Delta \mathbf{W}$  时  $\tilde{L}(\mathbf{W})$  的变化  $\Delta \tilde{L}(\mathbf{W})$  即有

$$\begin{aligned} \Delta L(\mathbf{W}) &= L(\mathbf{W} + \Delta \mathbf{W}) - L(\mathbf{W}) \approx \langle \nabla_{\mathbf{w}} L(\mathbf{W}) | \Delta \mathbf{W} \rangle \\ &= \text{tr}[\nabla_{\mathbf{w}} \tilde{L}(\mathbf{W}) (\Delta \mathbf{W})^T] \end{aligned}$$

(参见式 7-48b) 和式 (7-47)) 式中  $\tilde{L}(\mathbf{W})$  的梯度可由式 (7-58) 和式 (7-56) 对比中得到，即有

$$\nabla_{\mathbf{w}} \tilde{L}(\mathbf{W}) = -\mathbf{W}^{-T} + \frac{1}{T} \sum_{t=1}^T \boldsymbol{\varphi}(\mathbf{Y}(t)) \mathbf{X}^T(t) \quad (7-79)$$

$\boldsymbol{\varphi}(\mathbf{Y}(t))$  按式 7-57 计算。如果定义

$$\varphi_i(y_i(t)) = -\hat{p}'_i(y_i(t)) / \hat{p}_i(y_i(t)), \quad i = 1 \sim N \quad (7-80)$$

则

$$\boldsymbol{\varphi}(\mathbf{Y}(t)) = [\varphi_1(y_1(t)), \varphi_2(y_2(t)), \dots, \varphi_N(y_N(t))]^T \quad (7-81)$$

这样得到

$$\Delta \tilde{L}(\mathbf{W}) \approx -\text{tr}[\mathbf{W}^{-T} (\Delta \mathbf{W})^T] + \text{tr}\left[\frac{1}{T} \sum_{t=1}^T \boldsymbol{\varphi}(\mathbf{Y}(t)) \mathbf{X}^T(t) (\Delta \mathbf{W})^T\right]$$

假设  $\mathbf{W} \mathbf{W}^{-1}$  等于一个  $N \times N$  维矩阵  $\boldsymbol{\beta}$  即

$$\Delta \mathbf{W} \mathbf{W}^{-1} = \boldsymbol{\beta} = (\beta_{ij}) \quad (7-82)$$

则有

$$\Delta \tilde{L}(\mathbf{W}) \approx -\text{tr}[\boldsymbol{\beta}^T] + \text{tr}\left[\left(\frac{1}{T} \sum_{t=1}^T \boldsymbol{\varphi}(\mathbf{Y}(t)) \mathbf{Y}^T(t)\right) \boldsymbol{\beta}^T\right]$$

如果用  $\hat{E}[\mathbf{B}(t)]$  表示  $\frac{1}{T} \sum_{t=1}^T \mathbf{B}(t)$ ， $\mathbf{B}(t)$  可以是矩阵、向量或标量，则得到

$$\Delta \tilde{L}(\mathbf{W}) \approx -\sum_{i=1}^N \beta_{ii} + \text{tr}[\hat{E}[\boldsymbol{\varphi}(\mathbf{Y}(t)) \mathbf{Y}^T(t)] \boldsymbol{\beta}^T] \quad (7-83)$$

$\hat{E}[\varphi(\mathbf{Y}(t))\mathbf{Y}^T(t)]$  是一个  $N \times N$  维矩阵 其第  $i$  行第  $j$  列元素为  $E[\varphi_i(y_i(t))y_j(t)]$ 。这样即可求得

$$\Delta \tilde{L}(\mathbf{W}) \approx - \sum_{i=1}^N \beta_{ii} + \sum_{i=1}^N \sum_{j=1}^N \hat{E}[\varphi_i(y_i(t))y_j(t)]\beta_{ij} \quad (7-84)$$

ICA 的解  $\hat{\mathbf{W}}$  应是  $\tilde{L}(\mathbf{W})$  的一个极小值点。在此点上 如果  $\beta$  的各元素  $\beta_{ij}$  为充分小的非零值 而  $\Delta \tilde{L}(\hat{\mathbf{W}}) = 0$  则由式 (7-84) 可确定

$$\hat{E}[\varphi_i(y_i(t))y_j(t)] \Big|_{\mathbf{Y}(t)=\hat{\mathbf{W}}\mathbf{X}(t)} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \quad (7-85)$$

根据不同用途可由此基本公式导出其他计算公式。例如, 由于样本个数  $T$  有限, 即使式 (7-85) 得到满足 由其得到的  $\hat{\mathbf{W}}$  也不是 ICA 的理想解。在理想的情况下, 如果混合阵为  $\mathbf{A}$  则应有  $\mathbf{W} = \mathbf{A}^{-1}$ 。对于非理想情况可以设

$$\mathbf{W} = [\mathbf{I} + \boldsymbol{\varepsilon}]\mathbf{A}^{-1} \quad (7-86a)$$

其中  $\boldsymbol{\varepsilon} = (\varepsilon_{ij})$  描述了  $\mathbf{W}$  与理想解之间的相对差异。 $\varepsilon_{ii}$  诸项只影响各恢复源信号的尺度, 可暂时不考虑 而  $\varepsilon_{ij} (i \neq j)$  诸项显示了信号源  $j$  对信号源  $i$  的干扰。利用式 (7-85) 可以将诸  $\varepsilon_{ij} (i \neq j)$  近似计算出来, 从而可以估计出信号分离的质量。其计算思路如下所述。由

$\mathbf{Y}(t) = \mathbf{W}\mathbf{X}(t)$   $\mathbf{X}(t) = \mathbf{A}\mathbf{S}(t)$ , 按式 (7-86a) 得到

$$\mathbf{Y}(t) = [\mathbf{I} + \boldsymbol{\varepsilon}]\mathbf{S}(t) \quad (7-86b)$$

将此方程式按矩阵形式展开后得到

$$y_i(t) = S_i(t) + \sum_{l=1}^N \varepsilon_{il} S_l(t) \quad (7-86c)$$

将此式代入式 (7-85) 并且注意

$$\varphi_i(y_i(t)) = \varphi_i(S_i(t)) + \varphi'_i(S_i(t)) \sum_{l=1}^N \varepsilon_{il} S_l(t)$$

则当  $i \neq j$  时下式成立 (略去诸  $\varepsilon_{ij}$  的二次幂项):

$$\hat{E}[\varphi_i(S_i(t))S_j(t)] \approx - \sum_{l=1}^N \hat{E}[\varphi'_i(S_i(t))S_j(t)S_l(t)]\varepsilon_{il} - \sum_{l=1}^N \hat{E}[\varphi_i(S_i(t))S_l(t)]\varepsilon_{jl}$$

当  $T$  足够大时, 因为诸  $S_i(t)$  的平均值为 0 得到

$$\hat{E}[\varphi'_i(S_i(t))S_j(t)S_l(t)] \approx E[\varphi'_i(S_i(t))S_j(t)S_l(t)] = \begin{cases} E[\varphi'_i(S_i(t))S_j^2(t)], & l = j \\ 0, & l \neq j \end{cases}$$

$$\hat{E}[\varphi_i(S_i(t))S_l(t)] \approx E[\varphi_i(S_i(t))S_l(t)] = \begin{cases} E[\varphi_i(S_i(t))S_i(t)], & l = i \\ 0, & l \neq i \end{cases}$$

注意上列二式中  $\hat{E}[\cdot]$  与  $E[\cdot]$  相差一微小量, 此量乘以  $\varepsilon_{il}$  或  $\varepsilon_{jl}$  后成为二阶微小量, 可略去。这样, 当  $i \neq j$  时得到

$$E[\varphi_i(S_i(t))S_j(t)] \approx E[\varphi'_i(S_i(t))S_j^2(t)]\varepsilon_{ij} + E[\varphi_i(S_i(t))S_i(t)]\varepsilon_{ji} \quad (7-87)$$

这样, 每一对不同的  $(i, j)$  可以得到用上式描述的两个方程式 (例如,  $i = 1, j = 2$  和  $i = 2,$

$j = 1$ )。如果能对式中的各个  $E[\cdot]$  和  $\hat{E}[\cdot]$  估计出来, 就可以通过解联立方程求得相应的  $\epsilon_{ij}$  和  $\epsilon_{ji}$ 。这个问题将留到 7.6 节再讨论。现在只讨论如何按此思路用一个随机梯度迭代算法来求  $\mathbf{W}$  其目的是使得解  $\mathbf{W}$  满足式 (7-85)。

如果按照时序  $\nu = 1, 2, \dots$  送入观察向量  $\mathbf{X}(\nu)$  那么到任何一个时刻  $\nu = t$  共送入  $t$  个观察向量  $\mathbf{X}(1), \mathbf{X}(2), \dots, \mathbf{X}(t)$ 。若  $\hat{\mathbf{W}} = [\mathbf{I} + \boldsymbol{\epsilon}] \mathbf{A}^{-1}$ , 则可以将式 (7-87) 写成 (注意: 由于所有  $E[\cdot]$  恢复成  $E[\cdot]$  因此该式的约等号改成为等号。)

$$-\hat{E}[\varphi_i(S_i(t))S_j(t)] = \hat{E}[\varphi'_i(S_i(t))S_j^2(t)]\epsilon_{ij} + E[\varphi_i(S_i(t))S_i(t)]\epsilon_{ji}, \quad i \neq j$$
 即有

$$-\sum_{\nu=1}^t \varphi_i(S_i(\nu))S_j(\nu) = \left[ \sum_{\nu=1}^t \varphi'_i(S_i(\nu))S_j^2(\nu) \right] \epsilon_{ij} + \left[ \sum_{\nu=1}^t \varphi_i(S_i(\nu))S_i(\nu) \right] \epsilon_{ji}, \quad i \neq j$$

如果令  $\epsilon_{ij}$  和  $\epsilon_{ji}$  随  $t$  而改变 用  $\epsilon_{ij}(t)$  和  $\epsilon_{ji}(t)$  表示。并且注意到随着  $t$  的增加应该有

$$\sum_{\nu=1}^{t-1} \varphi_i(S_i(\nu))S_j(\nu) \rightarrow 0, \quad i \neq j$$

则上式可以近似表示为

$$-\varphi_i(S_i(t))S_j(t) = \Omega_{ij}(t)\epsilon_{ij}(t) + \Omega_{ji}(t)\epsilon_{ji}(t), \quad i \neq j \quad (7-88)$$

其中

$$\Omega_{ij}(t) = \sum_{\nu=1}^t \varphi'_i(S_i(\nu))S_j^2(\nu)$$

$$\Omega_{ji}(t) = \sum_{\nu=1}^t \varphi_i(S_i(\nu))S_i(\nu)$$

迭代算法总结如下。

(1) 首先随机设置初始参数  $\mathbf{W}(0)$ , 且令  $\Omega_{ij}(0) = 0$  和  $\Omega_{ji}(0) = 0 (i \neq j, i, j = 1 \sim N)$ 。

(2) 按时序  $\nu = 1, 2, \dots$  送入  $\mathbf{X}(\nu)$ 。对于任意  $\nu = t$  计算  $\mathbf{S}(t) = \mathbf{W}(t-1)\mathbf{X}(t)$ 。其中  $\tilde{\mathbf{S}}(t) = [\tilde{S}_1(t), \tilde{S}_2(t), \dots, \tilde{S}_N(t)]^T$  它是第  $t$  步时对于源信号的估计。

(3) 按下式计算  $t$  时刻的  $\Omega_{ij}(t)$  和  $\Omega_{ji}(t)$ :

$$\Omega_{ij}(t) = \lambda \Omega_{ij}(t-1) + \varphi'_i(\tilde{S}_i(t))\tilde{S}_j^2(t), \quad i \neq j$$

$$\Omega_{ji}(t) = \lambda \Omega_{ji}(t-1) + \varphi_i(S_i(t))S_i(t)$$

其中  $0 < \lambda \leq 1$  称为遗忘系数。

(4) 用下式解出各  $\epsilon_{ij}(t)$  和  $\epsilon_{ji}(t)$

$$\begin{cases} -\varphi_i(\tilde{S}_i(t))\tilde{S}_j(t) = \Omega_{ij}(t)\epsilon_{ij}(t) + \Omega_{ji}(t)\epsilon_{ji}(t), & i \neq j \\ -\varphi_j(\tilde{S}_j(t))\tilde{S}_i(t) = \Omega_{ji}(t)\epsilon_{ji}(t) + \Omega_{ij}(t)\epsilon_{ij}(t), & i \neq j \end{cases}$$

这样每一对  $\epsilon_{ij}(t), \epsilon_{ji}(t), i \neq j$  可由上述方程求解得到 从而可求得  $\boldsymbol{\epsilon}(t)$ 。

(5) 令  $\mathbf{W}(t) = [\mathbf{I} + \boldsymbol{\epsilon}(t)]\mathbf{W}(t-1)$ 。

(6) 计算下一节拍。

上列算法中的  $\lambda$  取值决定其自适应能力。若  $\lambda = 1$  则无自适应能力  $\lambda$  较小时自适应能力强而解的精度变差。此外 在 (3)、(4) 中需用到诸函数  $\varphi_i(\cdot)$  和  $\varphi'_i(\cdot)$  它们取决于信号源 pdf  $\hat{p}_i(S_i)$ 。如果事先不知道这些 pdf 则需在学习过程中加以确定。这留在 7.7 节中讨论。

### 7.4.5 ICA 的其他启发式学习算法

按照相对梯度或自然梯度构成的 ICA 学习算法已如前述。随机梯度和批处理的  $\mathbf{W}(k)$  迭代计算公式分别表述于式 (7-60) 和式 (7-62)。现设定一个  $N \times N$  维矩阵函数  $\mathbf{F}(\mathbf{Y}, \mathbf{W})$ ，即

$$\mathbf{F}(\mathbf{Y}, \mathbf{W}) = \mathbf{I} - \boldsymbol{\varphi}(\mathbf{Y})\mathbf{Y}^T \Big|_{\mathbf{Y}=\mathbf{W}\mathbf{X}} \quad (7-89)$$

则随机梯度算法可表示为

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \alpha(k)\mathbf{F}(\mathbf{Y}(k), \mathbf{W}(k))\mathbf{W}(k) \quad (7-90)$$

批处理算法可表示为

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \alpha(k)E[\mathbf{F}(\mathbf{Y}(t, k), \mathbf{W}(k))]\mathbf{W}(k) \quad (7-91)$$

其中

$$\hat{E}[\mathbf{F}(\mathbf{Y}(t, k), \mathbf{W}(k))] = \frac{1}{T} \sum_{t=1}^T \mathbf{F}(\mathbf{Y}(t, k), \mathbf{W}(k))$$

且

$$\lim_{T \rightarrow \infty} E[\mathbf{F}(\mathbf{Y}(t, k), \mathbf{W}(k))] = E[\mathbf{F}(\mathbf{Y}, \mathbf{W}(k))]$$

$\mathbf{F}(\mathbf{Y}, \mathbf{W})$  称为估计函数 (estimating function)。在批处理算法的情况下，如果迭代计算能够收敛到一个稳定平衡点  $\mathbf{W}$  这时应该有

$$E[\mathbf{F}(\mathbf{Y}(t), \mathbf{W})] = \mathbf{0} \quad (7-92)$$

其中  $\mathbf{Y}(t) = \hat{\mathbf{W}}\mathbf{X}(t)$ 。当  $T$  足够大时， $\hat{E}[\cdot]$  收敛于  $E[\cdot]$ 。这意味着下式成立：

$$E[\mathbf{F}(\mathbf{Y}, \hat{\mathbf{W}})] = \mathbf{0} \quad (7-93)$$

此式是估计函数应满足的必要条件。考察前面导出的式 (7-89) 目标函数满足此条件，其中  $E[\boldsymbol{\varphi}(\mathbf{Y})] = \mathbf{0}$  且  $E[\mathbf{Y}] = \mathbf{0}$ 。按照启发式的观点，可以构造出满足此条件的其他估计函数，并且通过实验来验证其有效性。下面列举出其中的一部分。

#### 1. 白化约束学习算法

式 (7-89) 给出的估计函数可分成偶对称部分  $\mathbf{F}_e(\cdot)$  和奇对称部分  $\mathbf{F}_o(\cdot)$  之和，即

$$\mathbf{F}(\mathbf{Y}, \mathbf{W}) = \mathbf{F}_e(\mathbf{Y}, \mathbf{W}) + \mathbf{F}_o(\mathbf{Y}, \mathbf{W})$$

其中

$$\mathbf{F}_e(\mathbf{Y}, \mathbf{W}) = \frac{1}{2}[\mathbf{F}(\mathbf{Y}, \mathbf{W}) + \mathbf{F}^T(\mathbf{Y}, \mathbf{W})] = \mathbf{I} - \frac{1}{2}[\boldsymbol{\varphi}(\mathbf{Y})\mathbf{Y}^T + \mathbf{Y}\boldsymbol{\varphi}^T(\mathbf{Y})] = \mathbf{F}_e^T(\mathbf{Y}, \mathbf{W})$$

$$\mathbf{F}_o(\mathbf{Y}, \mathbf{W}) = \frac{1}{2}[\mathbf{F}(\mathbf{Y}, \mathbf{W}) - \mathbf{F}^T(\mathbf{Y}, \mathbf{W})] = [\mathbf{Y}\boldsymbol{\varphi}^T(\mathbf{Y}) - \boldsymbol{\varphi}(\mathbf{Y})\mathbf{Y}^T] = -\mathbf{F}_o^T(\mathbf{Y}, \mathbf{W})$$

现在用  $\mathbf{I} - \mathbf{Y}\mathbf{Y}^T$  (这也是一个偶对称阵) 来替代  $\mathbf{F}_e(\mathbf{Y}, \mathbf{W})$ ，则可以构成另一种估价函数  $\mathbf{F}_1(\cdot)$ ，

$$\mathbf{F}_1(\mathbf{Y}, \mathbf{W}) = [\mathbf{I} - \mathbf{Y}\mathbf{Y}^T - \boldsymbol{\varphi}(\mathbf{Y})\mathbf{Y}^T + \mathbf{Y}\boldsymbol{\varphi}^T(\mathbf{Y})] \Big|_{\mathbf{Y}=\mathbf{W}\mathbf{X}} \quad (7-94)$$



易于验证,  $F_1(\cdot)$  满足估计函数的条件。如 7.2 节所述 如果满足  $E[I - YY^T] = 0$  的条件, 则  $Y$  的各分量之间不存在二阶相关性, 即称之为白化。因此, 按式 (7-94) 给出的  $F_1(\cdot)$  进行学习时即构成白化约束学习算法<sup>[18,16]</sup>

## 2. 解除 $W$ 的对角元素皆为 1 的约束

式 (7-89) 的估计函数  $F(\cdot)$  中第一项为单位阵  $I$ , 其隐含的作用是迫使恢复后的各源信号具有相同的均方差值 (等于 1)。这一约束可以放宽。如果令  $\Lambda_2$  是一个正定对角阵, 则可取估计函数  $F_2(\cdot)$  其定义为

$$F_2(Y, W) = \Lambda_2 - \phi(Y)Y^T \Big|_{Y=WX} \quad (7-95)$$

特别可以取  $\Lambda_2 = \text{diag}[\varphi_1(y_1)y_1, \varphi_2(y_2)y_2, \dots, \varphi_N(y_N)y_N]$ , 这可以改善迭代计算的收敛性<sup>[8]</sup>。

## 3. 加以白化约束又去除 $W$ 的对角约束

将 1、2 两项中所取的方法相结合可以形成  $F_3(\cdot)$  和  $F_4(\cdot)$  两种估计函数 即有

$$F_3(Y, W) = [\Lambda_3 - YY^T - \phi(Y)Y^T] \Big|_{Y=WX} \quad (7-96)$$

$$F_4(Y, W) = [\Lambda_4 - YY^T - \phi(Y)Y^T + Y\phi^T(Y)] \Big|_{Y=WX} \quad (7-97)$$

其中  $\Lambda_3 = \text{diag}[y_1^2 + \varphi_1(y_1)y_1, y_2^2 + \varphi_2(y_2)y_2, \dots, y_N^2 + \varphi_N(y_N)y_N]$

$$\Lambda_4 = \text{diag}[y_1^2, y_2^2, \dots, y_N^2]$$

## 4. 规格化学习算法

当采用式 (7-94) 的估计函数且按随机梯度算法进行迭代计算时, 计算公式可修正如下:

$$W(k+1) = W(k) + \alpha(k) \left\{ \frac{I - Y(k)Y^T(k)}{1 + \alpha(k)Y^T(k)Y(k)} + \frac{Y(k)\phi^T(Y(k)) - \phi(Y(k))Y^T(k)}{1 + \alpha(k)|\phi^T(Y(k))Y(k)|} \right\} W(k) \quad (7-98)$$

这一算法有如下优点。

(1) 可以提供“野点”(outlying) 保护。如果在运行过程中由于干扰或其他原因而出现  $Y(k)$  的模值突发增大的情况, 这称为野点。按照一般的算法可能造成很大误调, 而采用此算法时可以自动消除其影响。

(2) 当迭代计算已接近最优解时,  $|\phi^T(Y(k))Y(k)| \rightarrow 0$ ,  $Y^T(k)Y(k) \rightarrow N$  如果  $\alpha(k) < \frac{1}{N}$  则式 (7-98) 中所做的规格化处理几乎可略之不计。此算法即成为常规算法。

(3) 所增加的计算量很小。

若干实验结果证实, 这一算法是有效的, 见文献<sup>[16]</sup> 的实例。

### 7.4.6 迭代计算中步幅的自适应学习

和其他神经网络的学习算法一样, 迭代计算 (式 (7-90) 及式 (7-91)) 中步幅  $\alpha(k)$  的选择是至关重要的。 $\alpha(k)$  太小时收敛很慢  $\alpha(k)$  太大则会造成失调。在批处理的情况,  $\alpha(k)$  可选择为随  $k$  的增加而递减的函数, 如  $\alpha_0/k$ 。对于在线自适应的情况则很难作出选择, 一方面若  $\alpha(k)$  太小则跟不上环境的变化而达不到自适应目的, 另一方面  $\alpha(k)$  太大会引起过

调从而使分离误差增大。因而对于后者应取一种学习算法使得步幅可以随着环境变化程度的不同而作自适应调整。下面介绍该算法的一种<sup>[8,25,26]</sup>。首先 将式 (7-90) 写成

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \alpha(k)\mathbf{G}(k) \quad \mathbf{G}(k) = \mathbf{F}(\mathbf{Y}(k), \mathbf{W}(k))\mathbf{W}(k)$$

然后算出  $\mathbf{G}(k)$  的短期累计值  $\mathbf{G}(k)$  即有

$$\mathbf{G}(k) = (1 - \rho_2)\mathbf{G}(k-1) + \rho_2\mathbf{G}(k) \quad (7-99)$$

其中  $0 < \rho_2 < 1$ ,  $\rho_2$  越接近于 0 时累计作用越大。最后 计算出  $\alpha(k)$  之值 即

$$\alpha(k) = (1 - \rho_1)\alpha(k-1) + \rho_1\beta\psi(\mathbf{G}(k)) \quad (7-100)$$

其中  $0 < \rho_1 < 1, \beta > 0, \psi(\cdot)$  可用下式计算:

$$\psi(\hat{\mathbf{G}}(k)) = \tanh\left[\frac{1}{N^2} \sum_{i,j=1}^N (\hat{g}_{ij}(k))^2\right] \quad \text{或} \quad \frac{1}{N^2} \sum_{i,j=1}^N |\hat{g}_{ij}(k)| \quad (7-101)$$

其中  $\hat{g}_{ij}(k)$  是  $\mathbf{G}(k)$  的元素。 $\rho_1, \rho_2, \beta$  三个参数应依据所面临的问题而选择。

## 7.5 ICA 解的稳定性

为了说明稳定性的概念, 先讨论一个具有一维权参数变量  $w$  的目标函数  $l(y, w)$  其中  $y = wx, x$  是一个随机变量, 因此  $y$  也是随机变量。为了求一最佳参数  $\hat{w}$  使得  $E[l(y, \hat{w})]$  达到最小值 其中  $E[\cdot]$  表示对各种可能出现  $y$  的集合取平均) 则可以取下列迭代计算:

$$w(k+1) = w(k) + \alpha(k)f(y(k), w(k))$$

其中

$$f(y(k), w(k)) = -\left. \frac{\partial l(y, w)}{\partial w} \right|_{w=w(k)}$$

如果  $\hat{w}$  是所需的解, 则应该有

$$E[f(y, \hat{w})] = 0$$

$\hat{w}$  称为上列迭代计算的一个平衡点。但是, 一个平衡点不一定是一个稳定平衡点。假设

$$f'(y(k), w(k)) = -\left. \frac{\partial^2 l(y, w)}{\partial w^2} \right|_{w=w(k)}$$

那么只有在  $\hat{w}$  附近  $E[f'(y, w)]$  皆大于 0,  $\hat{w}$  才是一个稳定平衡点 图 7-2(a) 和 (b) 分别显示了稳定和非稳定平衡点的示例。对于前者, 当  $w$  稍偏离开  $\hat{w}$  时 迭代计算将使  $w$  返回  $\hat{w}$  对于后者 当  $w$  偏至  $\hat{w}$  左侧时 迭代计算将使  $w$  偏离  $\hat{w}$  越来越远。

### 7.5.1 ICA 的稳定解

对于 ICA 目标函数  $\tilde{L}(\mathbf{W})$  的变元  $\mathbf{W}$  是一个  $N \times N$  维矩阵。按照式 (7-90) 或式 (7-91) 进行的迭代计算可以求得一最优解  $\hat{\mathbf{W}}$  使  $\tilde{L}(\hat{\mathbf{W}})$  是一个极小值。为探讨解的稳定性, 以式 (7-90) 为例, 在其左右侧均右乘混合阵  $\mathbf{A}$  得到如下等价变化迭代计算式 (见 7.4.2 小节和式 (7-63)):

$$\mathbf{C}(k+1) = \mathbf{C}(k) + \alpha(k)\mathbf{F}(\mathbf{Y}(k), \mathbf{C}(k))\mathbf{C}(k) \quad (7-102)$$

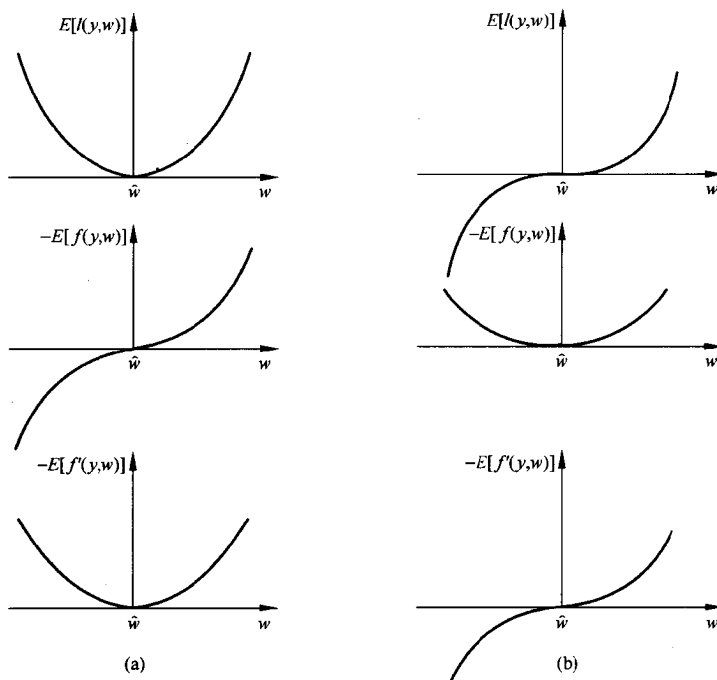


图 7-2 稳定和非稳定平衡点的示意图

注意 因为  $C(k) = W(k)A$ ,  $Y(k) = W(k)X(k)$ ,  $X(k) = AS(k)$  所以  $Y(k) = W(k)AS(k) = C(k)S(k)$ 。若  $W$  是所需解 则有  $C = WA = I$ ,  $Y(k) = S(k)$ 。这时下列方程得到满足 (参见式 (7-93)):

$$E[F(Y(k), C(k))] = E[F(Y(k), I)] = 0$$

其中  $F(Y(k), C(k))$  是一个  $N \times N$  维矩阵, 相应于  $L(W)$  的负梯度 (见 7.4 节) 可以看到,  $\hat{C} = I$  是它的一个平衡点。与一维情况相似, 还应进一步研究其是否稳定平衡点。为讨论方便 将  $F(Y(k), C(k))$  的  $N^2$  个元素  $f_{ij}$ ,  $i, j = 1 \sim N$  排列为一个  $N^2$  维列向量  $\theta$ ,

$$\theta = [f_{12}, f_{21}, f_{13}, f_{31}, \dots, f_{ij}, f_{ji}, \dots, f_{11}, f_{22}, \dots, f_{NN}]^T$$

这就是将  $i \neq j$  的各对  $f_{ij}, f_{ji}$  安置在  $\theta$  的前面  $N(N-1)$  个位置上 将  $f_{11}, \dots, f_{NN}$  安置在最后位置上。现在设  $W$  从  $\hat{W}$  偏离一个微小矩阵  $\Delta W$  相应地  $C$  从  $C = I$  偏离一个  $N \times N$  维相对变化微小阵  $\epsilon$ ,  $\epsilon = (\epsilon_{ij}) = \Delta WA$ ,  $C = C + \epsilon = I + \epsilon$  (见 7.4.1 小节的讨论),  $\epsilon$  的  $N^2$  个元素  $\epsilon_{ij}$  也排列成一个  $N^2$  维列向量  $\epsilon_V$ ,

$$\epsilon_V = [\epsilon_{12}, \epsilon_{21}, \epsilon_{13}, \epsilon_{31}, \dots, \epsilon_{ij}, \epsilon_{ji}, \dots, \epsilon_{11}, \epsilon_{22}, \dots, \epsilon_{NN}]^T$$

$\theta$  是  $Y$  和  $C$  的函数, 可表为  $\theta(Y, C)$ , 它其实就是矩阵  $F(Y, C)$  的向量形式, 因此  $E[\theta(Y, I)] = 0$ 。若  $C$  从  $I$  偏离到  $I + \epsilon$  则  $\theta$  从  $\theta(Y, I)$  偏离到  $\theta(Y, I + \epsilon)$ 。设  $\theta = \theta(Y, I + \epsilon) - \theta(Y, I)$  则

$$E[\Delta \theta]_{C=I} = -E[\theta(Y, I + \epsilon) - \theta(Y, I)] = K \epsilon_V \quad (7-103)$$

$K$  是一个  $N^2 \times N^2$  维二阶偏导矩阵, 与一维情况的二阶偏导数对应。在此多维的情况下,

当且只有当  $K$  的所有特征值大于 0 时  $W = I$  是稳定平衡点。下面讨论几种不同学习算法的稳定条件。

### 7.5.2 估计函数为白化约束函数 $F_1(\cdot)$ 时的稳定条件<sup>[16]</sup>

注意到  $E[\theta(Y, I)] = 0$  式(7-103)可写成

$$E[-\theta(Y, I + \varepsilon)] = K\varepsilon_v \quad (7-104)$$

$\theta(Y, I + \varepsilon)$  的原矩阵形式是  $F(Y, I + \varepsilon)$  它可以用式(7-89)计算。也可以用式(7-94)~式(7-97)给出的  $F_1(\cdot), F_2(\cdot), f_3(\cdot), f_4(\cdot)$  计算。

当估计函数为  $F_1(\cdot)$  时,按式(7-94)此原型可表示为

$$F_1(Y, I + \varepsilon) = \{I - YY^T - \varphi(Y)Y^T + Y\varphi^T(Y)\} \big|_{Y=(I+\varepsilon)S}$$

将  $E[-F_1(Y, I + \varepsilon)]$  的各元素求出,即能得到式(7-104)右侧的矩阵  $K$ 。为此将上式代入式(7-104)并把它分为两项。第一项是其中利用  $E[SS^T] = I$

$$E[(YY^T - I) \big|_{Y=(I+\varepsilon)S}] = E[(S + \varepsilon S)(S + \varepsilon S)^T - I] = \varepsilon + \varepsilon^T + \varepsilon\varepsilon^T$$

$\varepsilon\varepsilon^T$  相对于  $\varepsilon + \varepsilon^T$  可予忽略。因而此项的  $i$  行  $j$  列元素为  $\varepsilon_{ij} + \varepsilon_{ji}$ 。第二项是

$$E[(\varphi(Y)Y^T - Y\varphi^T(Y)) \big|_{Y=(I+\varepsilon)S}]$$

此项的  $i$  行  $j$  列元素为  $E[\varphi_i(y_i)y_j - y_i\varphi_j(y_j)]$ 。由  $Y = S + \varepsilon S$  得到

$$y_i = S_i + \sum_{l=1}^N \varepsilon_{il} S_l$$

$$\varphi_i(y_i) = \varphi_i(S_i) + \varphi'_i(S_i) \sum_{l=1}^N \varepsilon_{il} S_l \quad (\varphi'_i(S_i) \text{ 为 } \varphi_i(S_i) \text{ 的导数})$$

$$y_j = S_j + \sum_{r=1}^N \varepsilon_{jr} S_r$$

这样得到,当  $i \neq j$  时,

$$E[\varphi_i(y_i)y_j] = E[\varphi_i(S_i) \sum_{r=1}^N \varepsilon_{jr} S_r] + E[\varphi'_i(S_i) S_j \sum_{l=1}^N \varepsilon_{il} S_l] + E[\varphi_i(S_i) S_j] + O(\|\varepsilon\|)$$

其中最后一项较各  $\varepsilon_{ij}$  为可忽略的微少量;由于  $S_i$  和  $S_j$  相互独立且均值为 0 所以倒数第二项为 0 而前两项分别为  $E[\varphi_i(S_i)S_i]\varepsilon_{ji}$  和  $E[\varphi'_i(S_i)S_j^2]\varepsilon_{ij}$ 。因此,

$$E[\varphi_i(y_i)y_j] = E[\varphi_i(S_i)S_i]\varepsilon_{ji} + E[\varphi'_i(S_i)S_j^2]\varepsilon_{ij}$$

类似可求得

$$E[-y_i\varphi_j(y_j)] = -E[\varphi_j(S_j)S_j]\varepsilon_{ij} - E[\varphi'_j(S_j)S_i^2]\varepsilon_{ji}$$

将上述一、二两项的  $i$  行  $j$  列元素合并即得  $E[-F_1(Y, I + \varepsilon)]$  的第  $i$  行  $j$  列元素  $f_{ij}$  当  $i \neq j$  时为

$$f_{ij} = \{1 + E[\varphi'_i(S_i)S_j^2] - E[\varphi_j(S_j)S_j]\}\varepsilon_{ij} + \{1 - E[\varphi'_j(S_j)S_i^2] + E[\varphi_i(S_i)S_i]\}\varepsilon_{ji}$$

同样可求得其  $j$  行  $i$  列元素  $f_{ji}$  为

$$f_{ji} = \{1 + E[\varphi'_j(S_j)S_i^2] - E[\varphi_i(S_i)S_i]\}\varepsilon_{ji} + \{1 - E[\varphi'_i(S_i)S_j^2] + E[\varphi_j(S_j)S_j]\}\varepsilon_{ij}$$

当  $i = j$  时,由于  $E[\varphi_i(y_i)y_i - y_i\varphi_i(y_i)] = 0$  因此  $f_{ii} = 2\varepsilon_{ii}, i = 1 \sim N$ 。

注意,当  $i \neq j$  时,  $S_i$  与  $S_j$  相互独立且其均方值皆等于 1,故有  $E[\varphi'_i(S_i)S_j^2] = E[\varphi'_i(S_i)]E[S_j^2] = E[\varphi'_i(S_i)]$  类似地  $E[\varphi'_j(S_j)S_i^2] = E[\varphi'_j(S_j)]$ 。这样,一对  $f_{ij}, f_{ji}$  和

一对  $\epsilon_{ij}, \epsilon_{ji}$  之间的关系可以写成下列形式：

$$\begin{bmatrix} f_{ij} \\ f_{ji} \end{bmatrix} = \mathbf{F}_{ij} \begin{bmatrix} \epsilon_{ij} \\ \epsilon_{ji} \end{bmatrix} \quad (7-105a)$$

$$\mathbf{F}_{ij} = \mathbf{D} \begin{bmatrix} 2 & 0 \\ \xi_i - \xi_j & \eta_i + \eta_j \end{bmatrix} \mathbf{D}^T, \quad \mathbf{D} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} \quad (7-105b)$$

$$\xi_i = E[\varphi'_i(S_i) + S_i \varphi_i(S_i)], \quad \eta_i = E[\varphi'_i(S_i) - S_i \varphi_i(S_i)] \quad (7-105c)$$

$\xi_j$  与  $\eta_j$  的定义和  $\xi_i, \eta_i$  相同。

可以看到,  $\xi_i, \eta_i$  等取决于信号源的 pdf, 对于一定的信号源和一定的函数  $\varphi_i(\cdot)$  它们是一些常数。因此一对  $f_{ij}, f_{ji}$  只取决于相应的一对  $\epsilon_{ij}, \epsilon_{ji}$  而与其他  $\epsilon_{kl}, k \neq i$  或  $l \neq j$  无关。这样式 (7-104) 中的二阶偏导矩阵  $\mathbf{K}$  是一个具有下列形式的方块对角阵：

$$\mathbf{K} = \begin{bmatrix} \mathbf{F}_{12} & & & & & \\ & \mathbf{F}_{34} & & & & \\ & & \ddots & & & \\ & & & \mathbf{F}_{ij} & & \\ & & & & \ddots & \\ & 0 & & & & 2 \\ & & & & & & 2 \\ & & & & & & & \ddots \\ & & & & & & & & 2 \end{bmatrix} \quad (7-106)$$

即  $\mathbf{K}$  的前  $N(N-1)$  个对角线位置上是  $N(N-1)/2$  个  $2 \times 2$  维方阵  $\mathbf{F}_{ij}, 1 \leq i < j \leq N$ , 其最后  $N$  个对角元素为 2。由式 (7-105b) 可知各  $\mathbf{F}_{ij}$  的特征值是 2 和  $\eta_i + \eta_j$ 。这样为了保证  $\mathbf{K}$  的所有特征值皆大于 0, 下列条件应成立：

$$\eta_i + \eta_j > 0, \quad 1 \leq i < j \leq N \quad (7-107)$$

可以用此条件作一些初步的讨论。假设对任何  $i$  取函数  $\varphi_i(\cdot)$  为  $\varphi_i(S_i) = S_i^3$  则按照式 (7-105c) 得到

$$\eta_i = 3E[S_i^2] - E[S_i^4] = 3 - E[S_i^4], \quad i = 1 \sim N$$

其中应用了  $E[S_i^2] = 1, \forall i$ 。对照 7.2 节式 (7-9) 并根据同样条件即得到  $\eta_i = -\mathcal{H}_i$ 。由此可知, 如果各  $\varphi_i(\cdot)$  皆取立方函数时, 各信号源的峰起度皆为负或最多包含一个峰起度为 0 的信号源 (或一个峰起度为正但绝对值小于其他峰起度绝对值的信号源) 时, ICA 的解是稳定的。相反, 如果有两个或更多个信号源具有高斯或超高斯 pdf 时, 用立方函数  $\varphi_i(\cdot)$  的 ICA 不能得稳定解, 因而是不可分的。在 7.5.4 小节中将进一步讨论这个问题。

### 7.5.3 估计函数为 $\mathbf{F}(\cdot)$ 时的稳定条件 [27]

当采用式 (7-89) 给出的估计函数  $\mathbf{F}(\cdot)$  时, 平衡点的稳定条件由下式给出：

$$(1 + \eta_i)(1 + \eta_j) > 1, \quad 1 \leq i < j \leq N \quad (7-108)$$

其中  $\eta_i$  和  $\eta_j$  的定义仍按照式 (7-105c)。此式的证明见文献 [27] 和 [8] 此处不赘。文献

[18]将采用  $F(\cdot)$  的此学习算法称为非对称情况，将前一小节采用  $F_1(\cdot)$  的学习算法称为对称情况。图 7-3 标出了这两种算法的稳定界。可以看到，在其他条件一致的情况下， $F_1(\cdot)$  的稳定范围大于  $F(\cdot)$  的稳定范围

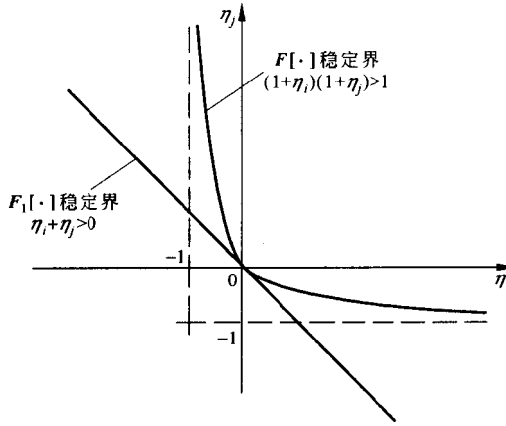


图 7-3  $F_1(\cdot)$  和  $F(\cdot)$  的稳定界

#### 7.5.4 ICA 稳定解的若干讨论

基于上列的推导，可以对 ICA 学习算法中解的稳定性问题作一些进一步探讨，列出如下。

(1) 不管学习算法取  $F(\cdot)$  还是  $F_1(\cdot)$  作为估计函数，只需满足  $\eta_i > 0, \forall i$ ，则解一定是稳定的，这是稳定的充分条件。因此针对一个特定的问题实施 ICA 算法时，为了实现正确分离，可归结为对每个信号源  $i$  正确选择  $\varphi_i(\cdot)$  使得  $\eta_i > 0$  的充分条件得到满足。

(2) 如果信号源  $i$  的函数  $\varphi_i(\cdot)$  中假设的该信号源 pdf  $\hat{p}_i(\cdot)$  (参见式 7-57)) 与其真实 pdf  $p_i(\cdot)$  完全一致时， $\eta_i$  达到其最大值  $\eta_{i\max}$ ，而  $\hat{p}_i(\cdot)$  与  $p_i(\cdot)$  不一致时， $\eta_i < \eta_{i\max}$ 。而且可以证明， $\eta_{i\max} \geq 0$ ，其中等号只有当 pdf 为高斯函数时成立。这样， $\eta_i$  提供了对于信号源  $i$  的 pdf 假设正确程度的定量性衡量。对于非高斯信号源  $i$ ，虽然其  $\eta_{i\max} > 0$ ，但是若  $\hat{p}_i(\cdot)$  与  $p_i(\cdot)$  相差过远以至于  $\eta_i \leq 0$ ，则仍不能实现正确分离。

(3) 如果信号源  $i$  的真实 pdf 为高斯函数，那么不管  $\varphi_i(\cdot)$  如何选择，总有  $\eta_i = 0$ 。此条与上一条都表明，当高斯信号源的个数超过 1 时，不能实现分离。这与前述论断一致。

(4) 如果预先知道各信号源均为超高斯的（其峰起度  $\mathcal{K}_i$  皆大于 0，见 7.2.1 小节）或均为次高斯的（其  $\mathcal{K}_i$  皆小于 0，亦见 7.2.1 小节），则可以用一种简单的  $\varphi_i(\cdot)$  选择实现信号分离。最简单的方法是令  $\varphi_i(S_i) = \beta_i S_i^3$ ，其中  $\beta_i = -\text{sgn}(\mathcal{K}_i)$ 。按照式 (7-105c) 可以求得其对应的  $\eta_i$  值为（引用式 7-9）

$$\eta_i = \beta_i \{3E[S_i^2] - E[S_i^4]\} = \text{sgn}(\mathcal{K}_i) \mathcal{K}_i = |\mathcal{K}_i|$$

这样，对于任何非高斯信号源皆有  $\eta_i > 0$ 。这说明，只要知道各信号源是超高斯的还是次高斯的，就足以实现分离，而无须知道其精确的 pdf 是什么。当然，如果不能预知各信号源

pdf 的特点或者是超高斯、次高斯俱存在的情况，则必须在学习中加以确定，详见 7.7 节的讨论。此外，文献 [8]，[28]，[27] 也指出，对于超高斯信号源取  $\varphi_i(S_i) = \alpha S_i \tanh(\gamma S_i)$  其中  $\alpha > 0, \gamma > 2$  对于次高斯信号源取  $\varphi_i(S_i) = \alpha S_i + S_i^3 (\alpha > 0)$  都可以实现分离，尽管它们不是最佳的选择。

## 7.6 用 ICA 实现源分离时解的精确度

本节讨论 ICA 得到的解  $\mathbf{Y}$  与源信号  $\mathbf{S}$  是否完全一致 如果不完全一致 二者的差异有多大，这些差异取决于哪些因素。按照 7.4.4 小节给出的思路，当 ICA 按批处理方式学习（例如式 (7-91) 给出的迭代计算公式） $\mathbf{WA}$  所收敛到的最优解  $\mathbf{WA}$  仍与理想最佳解  $\mathbf{I}$  有偏差 如果设  $\hat{\mathbf{W}}\mathbf{A} = \mathbf{I} + \boldsymbol{\varepsilon}$ ， $\boldsymbol{\varepsilon} = (\varepsilon_{ij})$  是一个  $N \times N$  维偏差矩阵。注意到（见式 (7-86a) 式 (7-86b) 和式 (7-86c)）

$$y_i(t) = S_i(t) + \sum_{l=1}^N \varepsilon_{il} S_l(t)$$

若  $\varepsilon_{ij} = 0, \forall i, j$  则  $y_i(t) = S_i(t), \forall i$  即恢复信号与源信号完全一致。若  $\varepsilon_{ij} \neq 0, i \neq j$ ，则二者不一致， $\varepsilon_{ij}$  越大 则  $S_j(t)$  的成分在  $y_i(t)$  中所占比重越大 即  $j$  信号对恢复后的  $i$  信号的干扰越严重。 $\varepsilon_{ij}, i \neq j$  称为污染 (contamination) 系数。下面将说明  $\varepsilon_{ij}, i \neq j$  是均值为 0 的随机变量，只要求出它的均方差值就能以其衡量 ICA 解的精确度，也只有用它才能衡量  $S_j(t)$  对  $S_i(t)$  的干扰（污染）。应指出 若  $\varepsilon_{ii} \neq 0, \forall i$ ，只会造成恢复本信号的幅度变化而不会造成信号之间的干扰，而且  $\varepsilon_{ii}$  值一般都很小，所以在讨论精确度时可以不予考虑。

为研究各  $\varepsilon_{ij}, 1 \leq i, j \leq N, i \neq j$ ，引用式 (7-87) 并且将其写成下列矩阵形式（为简化，将各信号的时间变量  $t$  略去）：

$$\mathbf{B}_{ij} = \mathbf{H}_{ij} \boldsymbol{\varepsilon}_{ij}, \quad 1 \leq i < j \leq N \quad (7-109a)$$

$$\left. \begin{aligned} \mathbf{B}_{ij} &= [\beta_{ij}, \beta_{ji}]^T = [\hat{E}[\psi_{ij}], \hat{E}[\psi_{ji}]]^T \\ \psi_{ij} &= -\varphi_i(S_i)S_j, \quad \psi_{ji} = -\varphi_j(S_j)S_i \end{aligned} \right\} \quad (7-109b)$$

$$\mathbf{H}_{ij} = \begin{bmatrix} E[\varphi'_i(S_i)]E[S_j^2] & E[\varphi_i(S_i)S_j] \\ E[\varphi_j(S_j)S_j] & E[\varphi'_j(S_j)]E[S_i^2] \end{bmatrix} \quad (7-109c)$$

$$\boldsymbol{\varepsilon}_{ij} = [\varepsilon_{ij}, \varepsilon_{ji}]^T \quad (7-109d)$$

注意，在由式 (7-87) 导出此式时，由于  $E[\cdot]$  表示对信号源的全集合取平均值且  $S_i$  与  $S_j$  相互统计独立 ( $i \neq j$ ) 所以  $E[\varphi'_i(S_i)S_j^2] = E[\varphi'_i(S_i)]E[S_j^2]$ 。而且  $E[S_j^2] = 1$  此项即等于  $E[\varphi'_i(S_i)]$ 。 $2 \times 2$  维矩阵  $\mathbf{H}_{ij}$  的 4 个元素都是集合平均值，因此它们不是随机变量而是取决于各  $S_i$  统计特性（即其真实 pdf 的常数。由于  $E[\cdot]$  表示批平均值 即

$$\hat{E}[g] = \frac{1}{T} \sum_{t=1}^T g(t) \quad (7-110)$$

若  $g$  为随机变量  $E[g]$  也是随机变量。因此 2 维列向量  $\mathbf{B}_{ij}$  的两个分量  $\beta_{ij}$  和  $\beta_{ji}$  都是随机变量。这样  $\boldsymbol{\varepsilon}_{ij}$  的两个分量  $\varepsilon_{ij}$  和  $\varepsilon_{ji}$  也是随机变量。显然，若  $\beta_{ij} = \beta_{ji} = 0$  且  $\mathbf{H}_{ij}$  非奇，则

$\epsilon_{ij} = \epsilon_{ji} = 0$ 。若  $\beta_{ij}$  和  $\beta_{ji}$  不为 0 则可按以下推导出  $\epsilon_{ij}$  和  $\epsilon_{ji}$  的均方差值。

首先 因为  $E[\psi_{ij}] = -E[\varphi_i(S_i)S_j] = -E[\varphi_i(S_i)]E[S_j] = 0$  ( $S_i, S_j$  相互独立且  $S_j$  之平均值为 0) 同样  $E[\psi_{ji}] = 0$  得到  $\mathbf{B}_{ij}$  的平均值为

$$E[\mathbf{B}_{ij}] = \mathbf{0}, \quad 1 \leq i < j \leq N \quad (7-111)$$

若  $\mathbf{H}_{ij}$  为非奇 则

$$E[\mathbf{e}_{ij}] = \mathbf{0}, \quad 1 \leq i < j \leq N \quad (7-112)$$

其次 用一般概率理论 例如参见文献[29] 可以求得  $\mathbf{B}_{ij}$  的协方差阵为

$$E[\mathbf{B}_{ij}\mathbf{B}_{ij}^T] = \frac{1}{T}\mathbf{G}_{ij}, \quad 1 \leq i < j \leq N \quad (7-113a)$$

$$\mathbf{G}_{ij} = \begin{bmatrix} E[\psi_{ij}\psi_{ij}] & E[\psi_{ij}\psi_{ji}] \\ E[\psi_{ji}\psi_{ij}] & E[\psi_{ji}\psi_{ji}] \end{bmatrix} = \begin{bmatrix} E[\varphi_i^2(S_i)]E[S_j^2]E[\varphi_i(S_i)S_i] & E[\varphi_i(S_i)S_j] \\ E[\varphi_j(S_j)S_i]E[\varphi_i(S_i)S_i] & E[\varphi_j^2(S_j)]E[S_i^2] \end{bmatrix} \quad (7-113b)$$

这样  $\mathbf{e}_{ij}$  的协方差阵  $\mathbf{\Sigma}_{ij} = E[\mathbf{e}_{ij}\mathbf{e}_{ij}^T]$  可由下式求出。由式 7-109a) 得

$$E[\mathbf{B}_{ij}\mathbf{B}_{ij}^T] = E[\mathbf{H}_{ij}\mathbf{e}_{ij}\mathbf{e}_{ij}^T\mathbf{H}_{ij}^T] = \mathbf{H}_{ij}\mathbf{\Sigma}_{ij}\mathbf{H}_{ij}^T$$

若  $\mathbf{H}_{ij}$  非奇 即得到

$$\mathbf{\Sigma}_{ij} = \frac{1}{T}\mathbf{H}_{ij}^{-1}\mathbf{G}_{ij}\mathbf{H}_{ij}^{-T}, \quad 1 \leq i < j \leq N \quad (7-114)$$

此外 若  $\mathbf{H}_{ij}$  非奇, 根据式 7-111) 和式 7-109a) 得到

$$E[\mathbf{e}_{ij}] = \mathbf{H}_{ij}^{-1}E[\mathbf{B}_{ij}] = \mathbf{0} \quad (7-115)$$

式( 7-115) 表明 ICA 解是一个无偏估计。设  $\mathbf{\Sigma}_{ij}$  的两个对角元素是  $\sigma_{ai}^2$  和  $\sigma_{ji}^2$  它们分别表示  $j$  信号源对  $i$  信号源干扰的均方值和  $i$  信号源对  $j$  信号源干扰的均方值, 只要求得了它们 就可以估计出 ICA 解的精度, 为此这要求  $\mathbf{\Sigma}$ 。如果假设

$$\rho_i = \frac{E[\varphi_i(S_i)S_i]}{\sqrt{E[\varphi_i^2(S_i)]}}, \quad \lambda_i = \frac{E[\varphi_i(S_i)S_i]}{E[\varphi_i'(S_i)]}, \quad \forall i \quad (7-116)$$

则由式 7-109c) 可将  $\mathbf{H}_{ij}$  表示为

$$\mathbf{H}_{ij} = \mathbf{D}_{ij} \begin{bmatrix} \rho_i/\lambda_i & \rho_i \\ \rho_j & \rho_j/\lambda_j \end{bmatrix}, \quad \mathbf{D}_{ij} = \begin{bmatrix} \sqrt{E[\varphi_i^2(S_i)]} & 0 \\ 0 & \sqrt{E[\varphi_j^2(S_j)]} \end{bmatrix} \quad (7-117)$$

而由式 7-113b) 可将  $\mathbf{G}_{ij}$  表示为

$$\mathbf{G}_{ij} = \mathbf{D}_{ij} \begin{bmatrix} 1 & \rho_i\rho_j \\ \rho_j\rho_i & 1 \end{bmatrix} \mathbf{D}_{ij}^T \quad (7-118)$$

将式 7-117) 和式 7-118) 代入式 7-114) 得到

$$\mathbf{\Sigma}_{ij} = \frac{1}{T} \begin{bmatrix} \rho_i/\lambda_i & \rho_i \\ \rho_j & \rho_j/\lambda_j \end{bmatrix}^{-1} \begin{bmatrix} 1 & \rho_i\rho_j \\ \rho_j\rho_i & 1 \end{bmatrix} \begin{bmatrix} \rho_i/\lambda_i & \rho_j \\ \rho_i & \rho_j/\lambda_j \end{bmatrix}^{-1} \quad (7-119)$$

由此即可求得  $\mathbf{\Sigma}_{ij}$  见文献[17]。现在只讨论一种最简单的情况 —— 所有信号源具有相同 pdf 从而  $\rho_i = \rho_j = \rho, \lambda_i = \lambda_j = \lambda, \forall i, j$ 。这时可以导出<sup>[17]</sup>



$$\Sigma_{ij} = \frac{\lambda^2}{T(1-\lambda^2)^2\rho^2} \begin{bmatrix} 1 + \lambda^2 - 2\lambda\rho^2 & \rho^2(1 + \lambda^2) - 2\lambda \\ \rho^2(1 + \lambda^2) - 2\lambda & 1 + \lambda^2 - 2\lambda\rho^2 \end{bmatrix} \quad (7-120)$$

这样，

$$\sigma_{\text{eij}}^2 = \sigma_{\text{ej}i}^2 = \frac{\lambda^2(1 + \lambda^2 - 2\lambda\rho^2)}{T(1 - \lambda^2)^2\rho^2}, \forall i, j \quad (7-121)$$

根据此式可以作进一步讨论。

(1) 按照式(7-80),  $\varphi(\cdot) = -\hat{p}'(\cdot)/\hat{p}(\cdot)$  其中  $\hat{p}(\cdot)$  是假设的源信号  $i$  的 pdf。如果其真实 pdf 为  $p_i(\cdot)$  设  $\varphi_i(\cdot) = -p'_i(\cdot)/p_i(\cdot)$  那么对于任何函数  $g(\cdot)$  下式成立(只需满足  $S \rightarrow \pm\infty$  时,  $g(s) \rightarrow 0$ )

$$E[g(S_i)\varphi_i^*(S_i)] = E[g'(S_i)]$$

这样就得到 当  $\varphi_i(S_i) = \varphi_i^*(S_i)$  时，

$$E[\varphi_i'(S_i)] = E[\varphi_i(S_i)\varphi_i^*(S_i)] = E[\varphi_i^2(S_i)]$$

$$E[\varphi_i(S_i)S_i] = E[\varphi_i^*(S_i)S_i] = E[(S_i)'] = 1$$

由此可得

$$\lambda_i = \rho_i^2, \quad \varphi_i(S_i) = \varphi_i^*(S_i) \quad (7-122)$$

如果满足条件  $\varphi(\cdot) = \varphi_i^*(\cdot) = \varphi^*(\cdot), \forall i$  则依据式(7-122)和式(7-121)得到  $\sigma_{\text{eij}}^2$  和  $\sigma_{\text{eji}}^2$  之最小值为

$$(\sigma_{\text{eij}}^2)_{\min} = (\sigma_{\text{eji}}^2)_{\min} = \frac{\rho^2}{T(1 - \rho^4)} \quad (7-123)$$

其中

$$\rho^2 = \frac{1}{E[\{\varphi^*(S)\}^2]}$$

可以证明<sup>[17]</sup> 当  $\varphi_i(S_i) = \varphi_i^*(S_i)$  时,  $\sigma_{\text{eij}} = (\sigma_{\text{eij}})_{\min}$ , 即只有当假设的源信号 pdf,  $\hat{p}(\cdot)$  与真实 pdf,  $p_i(\cdot)$  一致时, 信号源之间的干扰达到最小值。此最小值由式(7-123)给出。

(2)  $\sigma_{\text{eij}}^2$  随训练集的容量  $T$  的增加而下降。当  $T \rightarrow \infty$  时,  $\sigma_{\text{eij}}^2 \rightarrow 0$ 。

(3)  $(\sigma_{\text{eij}}^2)_{\min}$  取决于源信号的真实 pdf  $p_i(S_i)$ , 下面举一个例子。若 pdf 为高斯函数, 即  $p_i(S_i) = (2\pi)^{-\frac{1}{2}} \exp[-S_i^2/2]$  则

$$\varphi_i^*(S_i) = -p'_i(S_i)/p(S_i) = S_i$$

则可求得(由式(7-116))

$$\rho_i^2 = \frac{\{E[\varphi_i^*(S_i)S_i]\}^2}{E[(\varphi_i^*(S_i))^2]} = 1$$

因而 对于高斯信号源,  $(\sigma_{\text{eij}}^2)_{\min} \rightarrow \infty$ 。这再一次证明了, 当信号源中有一个以上具有高斯分布时是不可分的。

(4)  $\varphi_i(S_i) = \beta_i S_i^3$  时解的精确度讨论。在 7.4.4 小节中已证明, 若所有信号源的概率分布相同(即其真实 pdf 相同)时, 只需令  $\beta_i = -\text{sgn}(\mathcal{K}_i)$  即可得稳定解,  $\mathcal{K}_i$  为峰起度。若 ICA 取此  $\varphi_i(\cdot)$  则可求得其  $\sigma_{\text{eij}}^2$  如下。首先将  $\varphi_i(S_i) = \beta_i S_i^3$  代入式(7-116)即求得

$$\rho_i^2 = \rho^2 = \frac{(E[S_i^4])^2}{E[S_i^6]}, \quad \lambda_i^2 = \lambda^2 = \frac{(E[S_i^4])^2}{9(E[S_i^2])^2}$$

将其代入式 7-121(注意  $E[S_i^2] = 1$ ) 即可得

$$\sigma_{\epsilon ij}^2 = \sigma_{\epsilon ji}^2 = \frac{1}{T} \cdot \frac{E[S_i^6]\{9 + (E[S_i^4])^2 - 6(E[S_i^4])^2/E[S_i^6]\}}{(3 - E[S_i^4])^2(3 + E[S_i^4])^2}$$

由 7.2.1 小节的式 (7-9) 可知 若  $E[S_i^2] = 1$  则信号源的峰起度  $\mathcal{K}_i = E[S_i^4] - 3$ 。由此可见,  $\sigma_{\epsilon ij}^2$  与  $\mathcal{K}_i^2$  成反变关系。还可看到, 如果信号源的真实 pdf 所相应的  $\varphi_i^*(\cdot)$  恰为  $\varphi_i^*(S_i) = (\pm)S_i^3$  则在上面已经证明

$$E[(\varphi_i^*(S_i))^2] = E[(\varphi_i^*(S_i))']$$

即有

$$E[S_i^6] = E[3S_i^2] = 3$$

这样 将其代入上列  $\sigma_{\epsilon ij}$  即得  $(\sigma_{\epsilon ij})_{\min}$  为

$$(\sigma_{\epsilon ij}^2)_{\min} = (\sigma_{\epsilon ji}^2)_{\min} = \frac{3}{T} \cdot \frac{1}{|\mathcal{K}_i| (3 + E[S_i^4])}$$

由此可以看到, 信号源峰起度的绝对值越大时, 解的精确度越高。

## 7.7 ICA 算法中信号源 pdf 的确定

ICA 算法取得成功的关键在于对源信号的 pdf,  $\hat{p}_i(S_i)$ , 作出正确的假设。只有这样, 才能得到学习中所必须的函数  $\varphi_i(\cdot) = -\hat{p}'_i(\cdot)/\hat{p}(\cdot)$  (见式 7-57)。所谓正确是指  $\hat{p}_i(S_i)$  与真实的 pdf,  $p_i(S_i)$ , 尽可能接近; 至少, 在不能做到二者严格一致的情况下, 应保证  $\hat{p}_i(S_i)$  与  $p_i(S_i)$  具有同样的超高斯或次高斯特性。如果各信号源具有相似的统计特性——均为超高斯或均为次高斯且能够通过采集得到各信号源的幅度——出现频率直方图大致估计出其真实 pdf 的形状, 这问题简单地归结为找到一个与真实 pdf 接近又便于计算的解析函数作为  $\hat{p}_i(\cdot)$  并由之求得  $\varphi_i(\cdot)$ 。 $\varphi_i(\cdot)$  在 ICA 的计算中起关键作用, 因此常称为核心函数 (Core function)。下面将介绍一些常用的  $\hat{p}_i(\cdot)$  及其相应的核心函数  $\varphi_i(\cdot)$ 。在实际中常遇到一种更复杂和困难的情况, 这时不但信号源的 pdf 为未知, 而且既包含超高斯的又包含次高斯的信号源。对于这种情况就必须在学习过程中将  $\hat{p}_i(\cdot)$  及相应  $\varphi_i(\cdot)$  的学习包含在内, 这一节将介绍两种学习方案。如果信号源的 pdf 虽然并不精确知道, 但知道所有信号源均具有相同的超高斯或次高斯特性时, 可选用一种统一的  $\hat{p}_i(\cdot)$  及相应  $\varphi_i(\cdot)$ , 一般可取得好效果。

### 7.7.1 ICA 常用的 $\hat{p}_i$ 及相应的 $\varphi_i(\cdot)$

#### 1. $\hat{p}_i(S_i)$ 取双曲正割函数的平方

$$\hat{p}_i(S_i) = \text{sech}^2(S_i) = \left( \frac{2}{e^{S_i} + e^{-S_i}} \right)^2$$

相应的核心函数  $\varphi_i(S_i)$  为

$$\varphi_i(S_i) = -\frac{\hat{p}'_i(S_i)}{\hat{p}_i(S_i)} = 2\left(\frac{e^{S_i} - e^{-S_i}}{e^{S_i} + e^{-S_i}}\right) = 2\tanh(S_i)$$

将变量  $S_i$  换成  $y_i$  后, 式(7-57)的向量  $\boldsymbol{\varphi}(\mathbf{Y}(t))$  可表示为  $2\tanh(\mathbf{Y}(t))$  其中

$$\tanh(\mathbf{Y}(t)) = [\tanh(y_1(t)), \tanh(y_2(t)), \dots, \tanh(y_N(t))]^T$$

这样, 各种迭代计算公式中的  $\boldsymbol{\varphi}(\mathbf{Y}(t))$  都可以用  $2\tanh(\mathbf{Y}(t))$  替代, 例如式(7-60)可写成

$$\Delta \mathbf{W}(k) = \alpha(k)[\mathbf{I} - 2\tanh(\mathbf{Y}(k))\mathbf{Y}^T(k)]\mathbf{W}(k)$$

此  $\hat{p}_i(S_i)$  为超高斯的, 即其  $\mathcal{H}_i > 0$ 。相应  $\varphi_i(\cdot)$  为双曲正切函数, 就是人工神经网络中最常用的神经元 Sigmoid 函数。ICA 的早期经典研究中正是借用人工神经网络的思路来解决 BSS 问题时用了这一函数, 并且用来解决语音信号这一类超高斯信号源的分离, 取得了好效果<sup>[2]</sup>。至今, 它仍是解决这一类问题的优选函数之一。

## 2. 修正的双曲正割函数的平方

$$\hat{p}_i(S_i) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{S_i^2}{2}\right) \text{sech}^2(S_i)$$

可以求得与之相应的核心函数为

$$\varphi_i(S_i) = -\frac{\hat{p}'_i(S_i)}{\hat{p}_i(S_i)} = S_i + \tanh(S_i)$$

这样  $\boldsymbol{\varphi}(\mathbf{Y}(t))$  可以用  $\mathbf{Y}(t) + \tanh(\mathbf{Y}(t))$  替代, 而式(7-60)可以写成

$$\Delta \mathbf{W}(k) = \alpha(k)[\mathbf{I} - \mathbf{Y}(k)\mathbf{Y}^T(k) - \tanh(\mathbf{Y}(k))\mathbf{Y}^T(k)]\mathbf{W}(k)$$

此  $\hat{p}_i(S_i)$  仍然是超高斯的。可以看到, 采用此核心函数时, 相应的学习算法与前述有白化约束的学习算法(式(7-96))相似。

## 3. 混合高斯函数 (MOG)

MOG 是 mixture of Gaussian 的缩写, 其一般形式为(见文献[19], 有关详细讨论亦见该文)

$$\hat{p}_i(S_i) = \frac{(1-a)}{\sigma_1\sqrt{2\pi}} \exp\left[-\frac{(S_i-\mu_1)^2}{2\sigma_1^2}\right] + \frac{a}{\sigma_2\sqrt{2\pi}} \exp\left[-\frac{(S_i+\mu_2)^2}{2\sigma_2^2}\right]$$

当  $a = 0.5, \sigma_1 = \sigma_2 = \mu_1 = \mu_2 = 1$  时, 可以求得此  $\hat{p}_i(S_i)$  之峰起度为  $\mathcal{H}_i = -1/2$ (见文献[19])。相应的核心函数为

$$\varphi_i(S_i) = -\frac{\hat{p}'_i(S_i)}{\hat{p}_i(S_i)} = S_i - \tanh(S_i)$$

相应于式(7-60)的学习算法为

$$\Delta \mathbf{W}(k) = \alpha(k)[\mathbf{I} - \mathbf{Y}(k)\mathbf{Y}^T(k) + \tanh(\mathbf{Y}(k))\mathbf{Y}^T(k)]\mathbf{W}(k)$$

此  $\hat{p}_i(S_i)$  为次高斯的。

## 4. 混合双曲正割函数的平方

$$\hat{p}_i(S_i) = \rho[\text{sech}^2(S_i + b_i) + \text{sech}^2(S_i - b_i)]$$

其中  $\rho$  是一个规格化系数, 使得  $\hat{p}_i(S_i)$  在  $(-\infty, +\infty)$  范围内的积分值等于 1。易于求得, 相应的核心函数为

$$\varphi_i(S_i) = -\frac{\hat{p}'_i(S_i)}{\hat{p}_i(S_i)} = -2\tanh(S_i) + 2\tanh(S_i + b_i) + 2\tanh(S_i - b_i)$$

学习算法为

$$\Delta \mathbf{W}(k) = \alpha(k) [\mathbf{I} + 2\tanh(\mathbf{Y}(k))\mathbf{Y}^T(k) - 2\tanh(\mathbf{Y}(k) + \mathbf{b})\mathbf{Y}^T(k) - 2\tanh(\mathbf{Y}(k) - \mathbf{b})\mathbf{Y}^T(k)]\mathbf{W}(k)$$

其中  $\mathbf{b} = [b_1, b_2, \dots, b_N]^T$ 。可以证明<sup>[5]</sup> 当  $b_i = 0$  时  $\hat{p}_i(S_i)$  与普通双曲正割函数平方相同，具有超高斯特性。当  $b_i > 1$  时  $\hat{p}_i(S_i)$  为次高斯的。在实用中可通过  $b_i = 0$  或 2 来改变其超高斯或次高斯特性。

## 5. 指数函数族

$$\hat{p}_i(S_i) = f_q(S_i) = \rho_q \exp[-|S_i|^q/q]$$

其中  $q$  可取任何大于 0 的实数， $\rho_q$  是一个保证  $\hat{p}_i(S_i)$  的积分值为 1 的规格化系数。可以看到，当  $q = 2$  时 这就是一个高斯函数 当  $q < 2$  时为次高斯的 当  $q > 2$  时为超高斯的。相应的核心函数为

$$\varphi_i(S_i) = \text{sgn}(S_i) |S_i|^{q-1}$$

### 7.7.2 用开关法学习核心函数<sup>[5]</sup>

当信号源中既包含超高斯的又包含次高斯的 pdf 且不知道它们各自的数量时，各  $\varphi_i(\cdot)$  必须在学习过程中加以确定。这一小节讨论用开关法。注意上一小节讨论的两种 pdf：修正的双曲正割函数的平方和 MOG 前者为超高斯的 后者为次高斯的。如果信号源中包括这两种统计特性时，可以将此二者的学习算法合并如下：

$$\Delta \mathbf{W}(k) = \alpha(k) [\mathbf{I} - \mathbf{Y}(k)\mathbf{Y}^T(k) - \mathbf{J}\tanh(\mathbf{Y}(k))\mathbf{Y}^T(k)]\mathbf{W}(k)$$

其中  $\mathbf{J} = \text{diag}[J_1, J_2, \dots, J_N]$ 。若第  $i$  信号源为超高斯的， $J_i = 1$ ；若为次高斯的，则  $J_i = -1$ 。由于在事先并不知道哪些信号源是超高斯或次高斯的也不知道其数量，因此必须有一种方法在学习过程中确定诸  $J_i$ ，它们称为开关信号（在超高斯和次高斯特性之间开关）。

确定诸  $J$  的方法之一是利用 7.5 节讨论的稳定条件。已知由式 (7-105c) 定义的系数  $\eta_i$  大于 0 是学习稳定的充分条件，即

$$\eta_i = E[\varphi'_i(S_i)] - E[S_i \varphi_i(S_i)] > 0, \forall i$$

在采用开关学习算法时，核心函数为

$$\varphi_i(S_i) = S_i + J_i \tanh(S_i)$$

将其代入上式，即得到稳定的充分条件是

$$\eta_i = E[1 + J_i \text{sech}^2(S_i)] - E[S_i^2 + J_i S_i \cdot \tanh(S_i)] > 0$$

注意到  $E[1] = 1, E[S_i^2] = 1$  则可得

$$\eta_i = J_i \{E[\text{sech}^2(S_i)] - E[S_i \cdot \tanh(S_i)]\} > 0$$

可以看到 为保证此式成立 开关信号  $J_i$  应该满足

$$J_i = \text{sgn}\{E[\text{sech}^2(S_i)] - E[S_i \cdot \tanh(S_i)]\}$$

由于在学习过程中不能得到真正的集合平均  $E[\cdot]$ ，也不能得到真正的源信号  $S_i$ ，只能以

下列近似式替代上式：

$$J_i = \text{sgn}\{E[\text{sech}^2(y_i)] - E[y_i \tanh(y_i)]\}$$

除了上述的开关函数外，还可以用 7.7.1 小节 4 所给出的混合双曲正割函数的平方。令其系数  $b$  在 0 和 2 之间开关，即可得超高斯和次高斯的特性。

### 7.7.3 用函数逼近法学习核心函数<sup>[17]</sup>

设有  $L$  个基函数  $\bar{\varphi}_l(S_i), l = 1 \sim L$  则  $\varphi_i(S_i)$  可以用它们来进行逼近，即令

$$\varphi_i(S_i) = \sum_{l=1}^L C_{il} \bar{\varphi}_l(S_i), \quad \forall i$$

问题是如何选择诸  $C_{il}$  使得  $\varphi_i(S_i)$  与真实核心函数  $\varphi_i^*(S_i)$  最接近。7.6 节中已经指明，如果各个  $\varphi_i(S_i)$  与  $\varphi_i^*(S_i)$  越接近，则各个  $2 \times 2$  维矩阵  $\mathbf{H}_y^{-1} \mathbf{G}_y \mathbf{H}_y^{-T}$  的对角元素越小（见式 (7-114)）这时 ICA 的解中各信号的相互干扰越小，即解的精度越高。文献 [17] 证明，对于一组固定的基函数  $\bar{\varphi}_l(S_i)$  如果其中包含  $\bar{\varphi}_1(S_i) = S_i$ ，则可由下列方程组解得一组最佳系数  $C_{il}^*, l = 1 \sim L, i = 1 \sim N$  使得各个  $\mathbf{H}_y^{-1} \mathbf{G}_y \mathbf{H}_y^{-T}$  的对角元素达到使用该组基函数时的最小值。这组方程是

$$\sum_{l=1}^L C_{il}^* E[\bar{\varphi}_l(S_i) \bar{\varphi}_m(S_i)] = E[\varphi_i^*(S_i) \bar{\varphi}_m(S_i)], \quad m = 1 \sim L \quad (7-124)$$

注意到  $E[\varphi_i^*(S_i) g(S_i)] = E[g'(S_i)]$  (见 7.6 节)，则  $E[\varphi_i^*(S_i) \bar{\varphi}_m(S_i)] = E[\bar{\varphi}_m'(S_i)]$ ，因此式 (5-124) 可写成

$$\sum_{l=1}^L C_{il}^* E[\bar{\varphi}_l(S_i) \bar{\varphi}_m(S_i)] = E[\bar{\varphi}_m'(S_i)], \quad m = 1 \sim L \quad (7-125)$$

在实际学习过程中，集合平均  $E[\cdot]$  难以实现，可用批平均  $\hat{E}[\cdot]$  替代，而  $S_i$  可以用  $y$  替代。这时  $C_{il}^*$  可由下列公式求得：

$$\sum_{l=1}^L C_{il}^* \hat{E}[\bar{\varphi}_l(y_i) \bar{\varphi}_m(y_i)] = \hat{E}[\bar{\varphi}_m'(y_i)], \quad m = 1 \sim L \quad (7-126)$$

文献 [17] 采用 7.7.1 小节 5 给出的指数函数族  $f_q(S_i) = \rho_q \exp(-|S_i|^q/q)$  来形成核心函数的基函数  $\psi_q(S_i) = -f_q'(S_i)/f_q(S_i)$ 。在该文献中使用了 3 种基函数，它们是

$$\bar{\varphi}_1(S_i) = \psi_1(S_i) = S_i$$

$$\bar{\varphi}_2(S_i) = \psi_{1.5}(S_i)$$

$$\bar{\varphi}_3(S_i) = \psi_4(S_i)$$

其中  $\psi_1(\cdot)$  相应于高斯函数的核心函数， $\psi_{1.5}(\cdot)$  相应于次高斯函数， $\psi_4(\cdot)$  相应于超高斯函数。为节省计算量，任何  $\varphi_i(S_i)$  可用下式逼近：

$$\varphi_i(S_i) = \sum_{l=1}^3 C_{il}^* \bar{\varphi}_l(S_i) = C_{i1}^* \psi_1(S_i) + C_{i2}^* \psi_{1.5}(S_i) + C_{i3}^* \psi_4(S_i)$$

$C_{i1}^* \sim C_{i3}^*$  可由式 (7-126) 解出。

## 7.8 盲信号抽取

盲信号抽取简记为 BSE(blind signal extraction的缩写)。BSE 与 BSS 的区别在于后者是在已知信号源个数为  $N$  的前提下从观察信号向量  $\mathbf{X}(t)$  中同时分离出  $N$  个信号源,即  $\mathbf{S}(t)$  而前者是逐个地从  $\mathbf{X}(t)$  中分离出各个信号源,甚至可以预先不知道信号源的个数。

BSE 的命题与 BSS 类似。设有  $T$  个  $N$  维列向量  $\mathbf{S}(t), t = 1 \sim T, \mathbf{A}$  为未知的  $M \times N$  维固定矩阵,  $M$  维观察向量  $\mathbf{X}(t) = \mathbf{A}\mathbf{S}(t)$ ; 需由已知的  $\mathbf{X}(t), t = 1 \sim T$  求得各未知的  $\mathbf{S}(t)$ 。为简化计,下面只讨论  $M = N$  的情况。实际上,下文讨论的算法对于  $M \neq N$  的情况也完全适用。

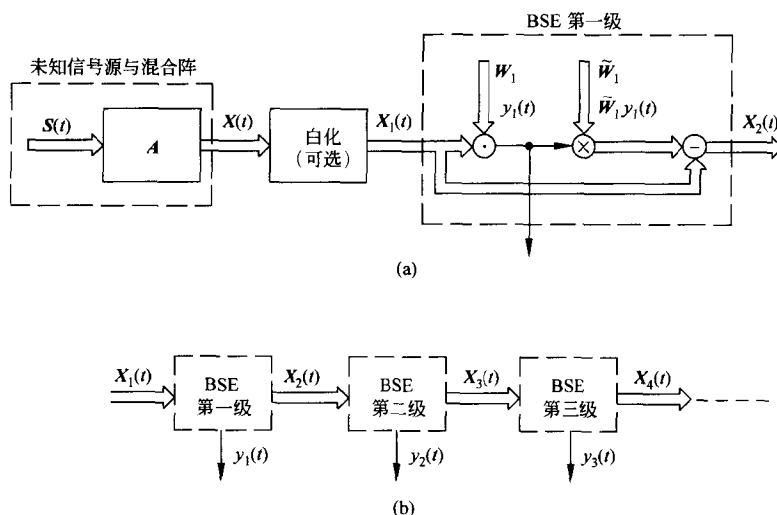


图 7-4 BSE 的计算流程图

BSE 的计算流程如图 7-4 所示。图 7-4(a) 所示是一个可选的白化模块和 BSE 的第一级模块。图 7-4(b) 所示是依次串接的 BSE 前三级模块,由于每一级模块的算法俱相同,所以只需要以第一级模块为例来讨论就足够了。白化模块的输入  $\mathbf{X}(t)$  和输出  $\mathbf{X}_1(t)$  都是  $N$  维列向量,后者是消除了前者各分量之间的二阶相关性(即白化)后得到的,可参阅 7.2.4 小节的 2 和 7.4.2 小节的 1。这是一个可选模块,选用它可以提高后续模块的学习效率(收敛速度)。BSE 的第一级模块包含两部分功能。第一部分是抽取功能。设  $\mathbf{W}_1$  是一个  $N$  维列向量,  $y_1(t) = \mathbf{W}_1^T \mathbf{X}_1(t)$  或写成点积形式  $y_1(t) = \mathbf{W}_1 \cdot \mathbf{X}_1(t)$ 。BSE 的目标是通过学习求得一最佳  $\mathbf{W}_1$ ,使得  $y_1(t)$  与  $\mathbf{S}(t)$  的某个分量  $S_i(t)$  尽可能接近。第二部分是紧缩(deflation)功能。设  $\tilde{\mathbf{W}}_1$  是另一个  $N$  维列向量,  $\mathbf{X}_2(t) = \mathbf{X}_1(t) - \tilde{\mathbf{W}}_1 y_1(t)$ 。这部分的目标是通过学习求得一最佳  $\tilde{\mathbf{W}}_1$  使得  $S_i(t)$  在  $\mathbf{X}_2(t)$  中的影响趋向于 0。这样经过 BSE 第一级模块处理后,抽取出一个与  $S_i(t)$  相应的信号  $y_1(t)$  并且将源信号  $S_i(t)$  剔除后的白化观察向量  $\mathbf{X}_2(t)$  送到第二模块。第二模块则抽取出与另一源信号  $S_j(t)$  接近的  $y_2(t)$  并将

$S_j(t)$  影响剔除后得到的  $\mathbf{X}_3(t)$  送到第三模块。依此类推 可一直做到第  $N$  模块为止。如果事先不知道源信号的个数，则必须设立一个终止阈值，当某个  $\mathbf{X}_m(t)$  的平均模值低于此阈值时就不必再作下一步抽取了。下面分别讨论  $\mathbf{W}_1$  和  $\tilde{\mathbf{W}}_1$  的学习算法。

### 7.8.1 $\mathbf{W}_1$ 的学习

首先 对于一定的  $\mathbf{W}_1, y_1(\cdot)$  与  $\mathbf{X}_1(t)$  具有下列关系：

$$y_1(t) = \mathbf{W}_1 \cdot \mathbf{X}_1(t) = \sum_{i=1}^N w_{1i} x_{1i}(t), \quad t = 1 \sim T \quad (7-127)$$

为使  $y_1(t)$  逼近某个  $S_i(t)$ ，采取下列目标函数：

$$L(\mathbf{W}_1) = \frac{-1}{4} |\bar{\mathcal{K}}(y_1)| = \frac{-1}{4} \left| \frac{\mathcal{K}(y_1)}{m_2^2(y_1)} \right| = \frac{-1}{4} \left| \frac{m_4(y_1)}{m_2^2(y_1)} - 3 \right| \quad (7-128)$$

其中  $m_4(y_1) = E[y_1^4]$ ,  $m_2(y_1) = E[y_1^2]$  分别是  $y_1$  的四阶和二阶矩。  $\mathcal{K}(y_1) = m_4(y_1) - 3m_2^2(y_1)$  是  $y_1$  的峰起度 参见 7.2.1 小节式 (7-9))。  $\bar{\mathcal{K}}(y_1)$  称为  $y_1$  的归一化峰起度。若能找到一个最优的  $\mathbf{W}_1$  使得  $L(\mathbf{W}_1)$  达到极小值 则  $y_1$  的归一化峰起度的绝对值达到极大。可以证明<sup>[8,30]</sup> 若有某个  $\mathbf{W}_1$  使  $L(\mathbf{W}_1)$  达到它的任何一个局部极小值，则  $y_1$  必然与某个源信号  $S_i$  相对应。为此可以从一个随机初值  $\mathbf{W}_1(0)$  出发 按节拍  $k = 0, 1, \dots$  进行迭代计算 直至收敛到一最佳值。由于  $L(\mathbf{W}_1)$  中使用的全集合平均  $E[\cdot]$  难以操作，在实际中以使用批平均的目标函数  $\tilde{L}(\mathbf{W}_1)$  代替之：

$$\tilde{L}(\mathbf{W}_1) = -\frac{1}{4} \left| \frac{\tilde{m}_4(y_1)}{\tilde{m}_2^2(y_1)} - 3 \right|, \quad \tilde{m}_l(y_1) = \frac{1}{T} \sum_{t=1}^T (y_1(t))^l = \hat{E}[y_1^l] \quad (7-129)$$

这样，可以按下列公式进行迭代计算：

$$\mathbf{W}_1(k+1) = \mathbf{W}_1(k) + \Delta \mathbf{W}_1(k) = \mathbf{W}_1(k) - \alpha(k) \nabla_{\mathbf{W}_1} \tilde{L}(\mathbf{W}_1) \big|_{\mathbf{W}_1 = \mathbf{W}_1(k)} \quad k = 0, 1, 2, \dots \quad (7-130)$$

其中  $\alpha(k)$  为步幅函数。当  $\mathbf{W}_1$  为迭代节拍  $k$  的函数  $\mathbf{W}_1(k)$  时，由式 (7-127)， $y_1(t)$  也是  $k$  的函数 可记为  $y_1(t, k) = \mathbf{W}_1(k) \mathbf{X}_1(t)$ 。将式 (7-129) 代入式 (7-130) 经过简单推导即得

$$\begin{aligned} \nabla_{\mathbf{W}_1} \tilde{L}(\mathbf{W}_1) \big|_{\mathbf{W}_1 = \mathbf{W}_1(k)} = \operatorname{sgn} \left[ \frac{\tilde{m}_4(y_1(t, k))}{\tilde{m}_2^2(y_1(t, k))} - 3 \right] & \left\{ \frac{1}{T} \sum_{t=1}^T \left[ \frac{\tilde{m}_4(y_1(t, k)) y_1(t, k)}{\tilde{m}_2^3(y_1(t, k))} - \right. \right. \\ & \left. \left. \frac{y_1^3(t, k)}{\tilde{m}_2^2(y_1(t, k))} \right] \mathbf{X}_1(t) \right\} \end{aligned} \quad (7-131)$$

其中

$$\tilde{m}_2(y_1(t, k)) = \frac{1}{T} \sum_{t=1}^T y_1^2(t, k), \quad \tilde{m}_4(y_1(t, k)) = \frac{1}{T} \sum_{t=1}^T y_1^4(t, k)$$

对于在线情况，特别是非平稳情况，可以将批处理迭代计算公式 (7-131) 修改为下列随机梯度计算公式，见文献 [31]。这时 每个节拍只输入一个  $\mathbf{X}_1$  并进行一次迭代计算。这样 迭代计算节拍  $k$  和输入时序  $t$  可统一用  $k$  表示。即对于  $k = 0, 1, 2, \dots$  输入  $\mathbf{X}_1(k), y_1(k) = \mathbf{W}_1(k) \cdot \mathbf{X}_1(k)$  权  $\mathbf{W}_1$  的迭代计算公式仍按式 (7-130) 而其中的批平均梯度  $\nabla_{\mathbf{W}_1} \tilde{L}(\mathbf{W}_1)$  变为按下式计算的随机梯度  $\nabla_{\mathbf{W}_1} l(\mathbf{W}_1)$

$$\nabla_{\mathbf{w}_1} l(\mathbf{W}_1) \big|_{\mathbf{w}_1 = \mathbf{w}_1(k)} = \text{sgn} \left[ \frac{\hat{m}_4(y_1(k))}{\hat{m}_2^2(y_1(k))} - 3 \right] \left[ \frac{\hat{m}_4(y_1(k)) y_1(k)}{\hat{m}_2^3(y_1(k))} - \frac{y_1^3(k)}{\hat{m}_2^2(y_1(k))} \right] \mathbf{X}_1(k) \quad (7-132a)$$

其中

$$\hat{m}_l(y_1(k)) = (1 - \eta) \hat{m}_l(y_1(k-1)) + \eta y_1^l(k-1), \quad l = 2, 4 \quad (7-132b)$$

$1 > \eta > 0$ ,  $\eta$  越接近于 0 则取平均的间隔越长。

### 7.8.2 $\tilde{\mathbf{W}}_1$ 的学习

与  $\mathbf{W}_1$  的学习相同,  $\mathbf{W}_1$  的学习算法可以分成批平均和在线随机梯度两种情况, 这里只介绍后者。已知  $\mathbf{X}_2(k) = \mathbf{X}_1(k) - \tilde{\mathbf{W}}_1(k) y_1(k)$  其中  $\mathbf{X}_1(k) = [x_{11}(k), x_{12}(k), \dots, x_{1N}(k)]^T$ ,  $\mathbf{X}_2(k) = [x_{21}(k), x_{22}(k), \dots, x_{2N}(k)]^T$ ,  $\tilde{\mathbf{W}}_1(k) = [\tilde{w}_{11}(k), \tilde{w}_{12}(k), \dots, \tilde{w}_{1N}(k)]^T$ 。现设置下列目标函数 ( $p$  为大于 1 的正整数):

$$\mathcal{L}(\tilde{\mathbf{W}}_1) \big|_{\tilde{\mathbf{w}}_1 = \tilde{\mathbf{w}}_1(k)} = \frac{1}{p} \sum_{i=1}^N |x_{2i}(k)|^p \quad (7-133)$$

可以证明<sup>[31]</sup> 若能求得最佳  $\tilde{\mathbf{W}}_1$  使  $\mathcal{L}[\mathbf{W}_1]$  达到最小值 则可使  $\mathbf{W}_1$  收敛到矩阵  $\mathbf{Q}\mathbf{A}$  的某一行向量 其中  $\mathbf{Q}$  是白化阵, 即  $\mathbf{X}_1 = \mathbf{Q}\mathbf{X} = \mathbf{Q}\mathbf{A}\mathbf{S}$ 。设在抽取中获得的  $y_1(k)$  逼近于  $\mathbf{S}(k)$  中的  $\mathbf{S}_i(k)$  则  $\tilde{\mathbf{W}}_1(k)$  的最佳值将逼近  $\mathbf{Q}\mathbf{A}$  矩阵的第  $i$  行的行向量, 这时  $\mathbf{X}_2(k)$  中不再包含  $\mathbf{S}_i(k)$  的影响。也就是通过紧缩步骤排除了源信号  $\mathbf{S}_i$ 。为求得使  $\mathcal{L}(\mathbf{W}_1)$  最小的  $\mathbf{W}_1$  可用下列迭代计算:

$$\tilde{\mathbf{W}}_1(k+1) = \tilde{\mathbf{W}}_1(k) + \Delta \tilde{\mathbf{W}}_1(k) = \tilde{\mathbf{W}}_1(k) - \tilde{\alpha}(k) \nabla_{\tilde{\mathbf{w}}_1} \mathcal{L}(\tilde{\mathbf{W}}_1) \big|_{\tilde{\mathbf{w}}_1 = \tilde{\mathbf{w}}_1(k)} \quad (7-134a)$$

$$\nabla_{\tilde{\mathbf{w}}_1} \mathcal{L}(\tilde{\mathbf{W}}_1) \big|_{\tilde{\mathbf{w}}_1 = \tilde{\mathbf{w}}_1(k)} = -y_1(k) \mathbf{G}(\mathbf{X}_2(k)) \quad (7-134b)$$

$$\mathbf{G}(\mathbf{X}_2(k)) = [g(x_{21}(k)), g(x_{22}(k)), \dots, g(x_{2N}(k))]^T, \quad g(x) = |x|^{p-1} \text{sgn}(x) \quad (7-134c)$$

经过第一级 BSE 抽取逼近  $\mathbf{S}_i(k)$  的  $y_1(k)$  经紧缩后得到的  $\mathbf{X}_2(k)$  不包含  $\mathbf{S}_i(k)$  的影响。将  $\mathbf{X}_2(k)$  送往第二级 BSE 抽取  $y_2(k)$  依次进行。这样,  $\mathbf{S}$  中的各个信号逐个得到恢复 直至抽取至某一级 (设第  $m-1$  级) 时  $\|\mathbf{X}_m(k)\| < \theta, \forall k, \theta$  是一个预置的小阈值。这表明所有信号源都已抽取掉了。  $m$  可等于  $N$ , 也可不等。后者相应于源信号数与观察信号数不同的情况。事实上, 采取 BSE 时根本无需预知信号源个数, 而可在学习中予以确定。此外 实验表明<sup>[31, 32]</sup> 如果用  $\hat{y}_1(k) = y_1(k) + \nu_1(k)$  替代  $y_1(k)$  来进行式 (7-132) 的计算 其中  $\nu_1(k)$  为随着  $k$  的增加而均方值逐渐减小的噪声, 则可使迭代计算的结果避免陷于不正确的局部极小点。

### 7.8.3 BSE 的特点

(1) BSE 按照峰起度  $\mathcal{K}_i$  绝对值的大小排序, 依次从大到小分离出各个源信号。

(2) 当信号源与大量高斯噪声或类高斯噪声信号相混合时, BSE 可以从中只分离出那些值得提取的信号, 即峰起度绝对值较大的信号。



(3) BSE 的学习算法较 BSS 简单。

(4) BSE 算法与盲解卷 ( BDC ) 问题密切相关<sup>[33,34]</sup>

## 7.9 盲解卷与盲均衡

### 7.9.1 概说

设有离散时间  $t$  为变量的  $N$  维源信号向量  $\mathbf{S}(t) = [S_1(t), S_2(t), \dots, S_N(t)]^T, t = \dots, -1, 0, 1, \dots$  各  $S_i(t)$  之间统计独立。相应的  $M$  维观察向量为  $\mathbf{X}(t) = [x_1(t), x_2(t), \dots, x_M(t)]^T, M \geq N$ 。二者之间关系由下式确定 (混合-卷积系统) :

$$\mathbf{X}(t) = \sum_{p=-\infty}^{+\infty} \mathbf{A}_p \mathbf{S}(t-p) \quad (7-135)$$

其中各  $\mathbf{A}_p$  为  $M \times N$  维实矩阵 称为混合-冲激响应阵。如果此混合-卷积系统是因果的 , 则有

$$\mathbf{X}(t) = \sum_{p=0}^{\infty} \mathbf{A}_p \mathbf{S}(t-p) \quad (\text{因果系统}) \quad (7-136)$$

如果混合-卷积系统是因果且有限冲激响应 ( FIR ) 的 , 则有

$$\mathbf{X}(t) = \sum_{p=0}^L \mathbf{A}_p \mathbf{S}(t-p), \quad L < \infty \quad (7-137)$$

提出的命题是 若已知  $\mathbf{X}(t), t = 0, 1, \dots$  在未知  $\mathbf{A}_p$  和  $\mathbf{S}(t)$  的条件下求  $\mathbf{S}(t), t = 0, 1, \dots$ 。若  $L = 0$  则每个  $\mathbf{X}(t)$  只与一个源向量  $\mathbf{S}(t)$  对应 , 则此命题退化为 BSS 问题。若  $L > 0$  则一个  $\mathbf{X}(t)$  将是多个  $\mathbf{S}(\cdot)$  的混合。在此情况下 , 称此命题为盲解卷 / 均衡问题。一些文献中将  $M > 1$  的情况称为盲解卷问题 , 将  $M = 1$  的情况称为盲均衡问题。本书即取此定义。

为从离散时间向量序列  $\mathbf{X}(t)$  求得源信号  $\mathbf{S}(t)$  可采取下列变换 (盲解卷系统) :

$$\mathbf{Y}(t) = [y_1(t), y_2(t), \dots, y_N(t)]^T = \sum_{p=-\infty}^{+\infty} \mathbf{W}_p \mathbf{X}(t-p) \quad (7-138)$$

其中各  $\mathbf{W}_p$  为  $N \times M$  维实矩阵。如果各  $\mathbf{W}_p$  选得恰当  $\mathbf{Y}(t)$  将趋于  $\mathbf{S}(t)$  (相差一个固定的延时) 。以上讨论针对的是信号源平稳 (即其统计特性不变) 且混合-冲激响应为恒定的情况。如上二者之一或二者皆随  $t$  而变化时  $\mathbf{W}_p$  也应随  $t$  而改变 , 这时可采取下列的自适应盲解卷 / 均衡算法 :

$$\mathbf{Y}(t) = \sum_{p=-\infty}^{+\infty} \mathbf{W}_p(t) \mathbf{X}(t-p) \quad (7-139)$$

为简化起见 , 下文只针对平稳情况的式 7-138 来讨论。自适应情况参见文献[8]。

设  $\mathbf{S}(t)$  的  $z$  变换为  $\mathbf{S}(z) = [S_1(z), S_2(z), \dots, S_N(z)]^T$  其中  $S_i(z) = \sum_{t=-\infty}^{+\infty} S_i(t) z^{-t}$ ,  $i = 1 \sim N$ 。设  $\mathbf{X}(t)$  的  $z$  变换为  $\mathbf{X}(z) = [X_1(z), X_2(z), \dots, X_M(z)]^T$  , 其中  $X_i(z) = \sum_{t=-\infty}^{+\infty} x_i(t) z^{-t}, i = 1 \sim M$ 。再设  $\mathbf{Y}(z) = [Y_1(z), Y_2(z), \dots, Y_N(z)]^T$  , 其中

$Y_i(z) = \sum_{t=-\infty}^{+\infty} y_i(t)z^{-t}$ 。设  $A(z)$  和  $W(z)$  是混合-卷积系统和盲解卷系统的  $z$  域传输函数，即

$$A(z) = \sum_{p=-\infty}^{+\infty} A_p z^{-p}, \quad W(z) = \sum_{p=-\infty}^{+\infty} W_p z^{-p} \quad (7-140)$$

则易于证明下列关系成立：

$$X(z) = A(z)S(z) \quad Y(z) = W(z)X(z) = W(z)A(z)S(z) \quad (7-141)$$

如果  $W(z)A(z) = \text{diag}[D_{11}(z), D_{22}(z), \dots, D_{NN}(z)]$  其中  $D_{ii}(z) = z^{-\Delta_i}$ ,  $i = 1 \sim N$ ,  $\Delta_i$  为某个整数。这时将有  $Y_i(z) = S_i(z)z^{-\Delta_i}$ ,  $i = 1 \sim N$  即  $y_i(t)$  与  $S_i(t)$  的区别仅在于前者较后者延迟了  $\Delta_i$  个时拍。盲解卷的目标即在于求得一种  $W(z)$  的学习算法使  $W(z)A(z)$  对角化。

## 7.9.2 频域盲解卷

频域盲解卷的基础是离散时域傅里叶变换 (DFT)。为此需限定  $S(t)$ 、 $X(t)$  和  $Y(t)$  的取值区间均为  $t = 0, 1, 2, \dots, T-1$ 。其中  $T = 2^n$ ,  $n$  是某个正整数 例如  $T$  可取值为  $\dots, 128, 256, \dots$  (若  $S(t)$  等的取值区间不能满足此条件时，可以在其取值区后方补若干 0 使之得到满足) 这样  $S(t)$ 、 $X(t)$ 、 $Y(t)$ ,  $t = 0 \sim T-1$  所对应的 DFT 为  $S(l) = [S_1(l), S_2(l), \dots, S_N(l)]^T$ ,  $X(l) = [X_1(l), X_2(l), \dots, X_M(l)]^T$ ,  $Y(l) = [Y_1(l), Y_2(l), \dots, Y_N(l)]^T$ ,  $l = 0 \sim T-1$ 。

$$S_i(l) = \sum_{t=0}^{T-1} S_i(t) \exp\left[-j \frac{2\pi}{T} tl\right], \quad i = 1 \sim N, l = 0 \sim T-1$$

$$X_i(l) = \sum_{t=0}^{T-1} x_i(t) \exp\left[-j \frac{2\pi}{T} tl\right], \quad i = 1 \sim M, l = 0 \sim T-1$$

$$Y_i(l) = \sum_{t=0}^{T-1} y_i(t) \exp\left[-j \frac{2\pi}{T} tl\right], \quad i = 1 \sim N, l = 0 \sim T-1$$

类似地  $A(z)$  与  $W(z)$  所相应的 DFT 是

$$A(l) = \sum_{p=0}^{T-1} A_p \exp\left[-j \frac{2\pi}{T} pl\right], \quad W(l) = \sum_{p=0}^{T-1} W_p \exp\left[-j \frac{2\pi}{T} pl\right], \quad l = 0 \sim T-1$$

$A(l)$  是  $M \times N$  维矩阵  $W(l)$  是  $N \times M$  维矩阵。诸 DFT 之间满足下列关系：

$$X(l) = A(l)S(l), \quad Y(l) = W(l)X(l) = W(l)A(l)S(l), \quad l = 0 \sim T-1$$

如果能使  $W(l)A(l) = \text{diag}[D_{11}(l), D_{22}(l), \dots, D_{NN}(l)]$ ,  $D_{ii}(l) = \exp\left[j \frac{2\pi}{T} l \cdot \Delta_i\right]$ ,  $i = 1 \sim N$ ,  $\Delta_i$  为任意的正整数 则  $X(l)$  与  $S(l)$  的各分量之间只相差一个延时量。

这样，频域盲解卷所要实现的是  $T$  个矩阵  $W(l)A(l)$ ,  $l = 0 \sim T-1$  对角化的问题。如果有多个有限时段： $t = k \sim k+T-1$ ,  $k = 0, 1, 2, \dots$ ，则可以将上述单个时段的表达式通过赋予时段标号  $k$  的方法成为一系列时段表达式，即令

$$S_k(t), X_k(t), Y_k(t) \text{ 对应于时段 } t = k \sim k+T-1$$

相应地 DFT 表示为  $S_k(l), X_k(l), Y_k(l)$  且满足

$$Y_k(l) = W_k(l)A_k(l)S_k(l), l = 0 \sim T-1, k = 0, 1, 2, \dots$$

这样,可以模仿 BSS 问题中的 ICA 算法,求得  $T$  个最佳频域最佳解  $\mathbf{W}(l)$ ,  $l = 0 \sim T-1$ , 使得  $\mathbf{W}(l)\mathbf{A}(l)$  为对角阵。为此,应规定  $M = N$  这时  $\mathbf{W}(l)$  和  $\mathbf{A}(l)$  都是  $N \times N$  维方阵。下面按式 (7-90) 给出的随机梯度算法并取式 (7-95) 给出的估计函数, 类比地得到下列频域盲解卷随机梯度算法:

$$\mathbf{W}_{k+1}(l) = \mathbf{W}_k(l) + \Delta \mathbf{W}_k(l), \quad l = 0 \sim T-1, k = 0, 1, 2 \dots \quad (7-142a)$$

$$\Delta \mathbf{W}_k(l) = \alpha(k) [\mathbf{A}_k(l) - \boldsymbol{\varphi}(\mathbf{Y}_k(l)) \mathbf{Y}_k^H(l)] \mathbf{W}_k(l) \quad (7-142b)$$

$$\mathbf{Y}_k(l) = [Y_{1k}(l), Y_{2k}(l), \dots, Y_{Nk}(l)]^T = \mathbf{W}_k(l) \mathbf{X}_k(l) \quad (7-142c)$$

$$\boldsymbol{\varphi}(\mathbf{Y}_k(l)) = [\varphi_{1l}(Y_{1k}(l)), \varphi_{2l}(Y_{2k}(l)), \dots, \varphi_{Nl}(Y_{Nk}(l))]^T \quad (7-142d)$$

$$\mathbf{A}_k(l) = \text{diag}[\boldsymbol{\varphi}(\mathbf{Y}_k(l)), \mathbf{Y}_k^H(l)] \quad (7-142e)$$

$$\mathbf{X}_k(l) = [X_{1k}(l), X_{2k}(l), \dots, X_{Nk}(l)]^T, \quad X_{ik}(l) = \sum_{t=k}^{k+T-1} x_i(t) \exp \left\{ -j \frac{2\pi}{T} tl \right\}, \quad i = 1 \sim N \quad (7-142f)$$

以上诸式中的  $\mathbf{W}_k(l)$  为复矩阵,  $X_{ik}(l), Y_{ik}(l)$  均为复数。式中的  $H$  符号为 Hermitian 的缩写, 表示“共轭转置”。式 (7-142d) 中的  $\boldsymbol{\varphi}(\cdot)$  与 BSS 问题中的  $\boldsymbol{\varphi}(\cdot)$  对应 (见式 7-81), 各  $\varphi_{il}(Y_{ik}(l))$  与  $\varphi_i(y_i(t))$  对应 (见式 7-80)。 $\varphi_i(\cdot)$  是根据源信号  $S_i$  的假设 pdf  $\hat{p}_i(\cdot)$  求出来的。那么各  $\varphi_{il}(\cdot)$  也应根据各  $S_i(l)$  的假设 pdf  $\hat{p}_{il}(\cdot)$  求出。注意到各  $S_i(l)$  为复数, 因此各  $\varphi_{il}(\cdot)$  也应是复数。有关频域盲解卷的讨论, 文献 [35] 有详细介绍, 文献 [17] 更有深入的理论分析。限于篇幅, 不在此详述。

利用 IDFT 易于将频域解得的  $\mathbf{W}(l)$  和  $\mathbf{Y}(l)$  转换到时域, 以得到时域解  $\mathbf{W}_p$  和  $\mathbf{Y}(t)$

$$\mathbf{W}_p = \frac{1}{T} \sum_{l=0}^{T-1} \mathbf{W}(l) \exp \left[ j \frac{2\pi}{T} pl \right]$$

$$\mathbf{Y}(t) = \frac{1}{T} \sum_{l=0}^{T-1} \mathbf{Y}(l) \exp \left[ j \frac{2\pi}{T} pl \right]$$

以上所述的 DFT 和 IDFT 都可用高效的 FFT 完成。

频域盲解卷的问题之一是计算量过大。可以看到, 与 BSS 相比, 其计算量增大为  $T$  倍 (还不包括 DFT 和 IDFT 的计算量), 而对于“拖尾”很长的冲激响应 (即式 7-137) 中的  $L$  很大),  $T$  必须足够大 (至少大于  $L$ ), 这时计算量可能过大。另外, 对于变化的环境,  $T$  过大也使此算法的自适应性能变差。

频域算法的另一个问题是当  $T$  较大时各  $S_i(l)$  趋向于正态分布, 这已在文献 [17] 中指出。按照 ICA 的观点, 正态分布信号源不具备可分性。虽然文献 [17]、[8]、[35] 都介绍了这种算法, 其效果和坚实的理论基础还需进一步研究。

### 7.9.3 盲均衡

设有一个源信号时间序列  $S(t), t = \dots, 0, 1, \dots$ , 任何两个不同时刻上的  $S(t)$  相互统计独立。 $S(t)$  通过一个  $z$  域函数为  $A(z)$  的信道后产生观察信号时间序列  $x(t), x(t)$  的各时间点值具有统计相关。为了从  $x(t)$  恢复  $S(t)$  可以模仿解决 BSS 问题的 ICA 算法, 首先

将  $x(t)$  变换为  $y(t)$

$$y(t) = \sum_{p=0}^L w_p(t)x(t-p) = \mathbf{W}^T(t)\mathbf{X}(t) \quad (7-143)$$

$\mathbf{W}(t) = [w_0(t), w_1(t), \dots, w_L(t)]^T$ ,  $\mathbf{X}(t) = [x(t), x(t-1), \dots, x(t-L)]^T$ ,  $L$  取决于  $A(z)$  延展的时域宽度。对于非平稳环境  $\mathbf{W}(t)$  随  $t$  而变化, 自适应于环境变化。仿照 ICA 的随机梯度算法 (见 7.4.5 小节的式 7-90) 按照迭代节拍  $k$  对  $\mathbf{W}(t)$  进行调整时可将信号时序  $t$  与迭代节拍  $k$  合并且统一用  $k$  表示。这时  $\mathbf{W}(t)$  和  $\mathbf{X}(t)$  都表示为  $\mathbf{W}(k)$  和  $\mathbf{X}(k)$ , 式 (7-143) 即成为  $y(k) = \mathbf{W}^T(k)\mathbf{X}(k)$ 。设  $\mathbf{Y}(k) = [y(k), y(k-1), \dots, y(k-L)]^T$ , 式 (7-90) 的迭代计算使得其中涉及的  $\mathbf{Y}(k)$  各分量统计独立, 因此仿照它构成的下列迭代计算也是使得相应的  $\mathbf{Y}(k)$  的各分量统计独立 从而令  $y(k)$  与  $S(k-Q)$  对应,  $Q$  是一整数。

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \alpha(k)\mathbf{F}(\mathbf{Y}(k), \mathbf{W}(k))\mathbf{W}(k) \quad (7-144)$$

其中  $\mathbf{F}(\cdot)$  是一个  $(L+1) \times (L+1)$  维矩阵。它可以取 7.4.5 小节给出的几种形式, 其中一种为

$$\mathbf{F}(\mathbf{Y}(k), \mathbf{W}(k)) = [\mathbf{A} - \boldsymbol{\varphi}(\mathbf{Y}(k))\mathbf{Y}^T(k)] \left| \begin{array}{l} y(k) = \mathbf{W}^T(k)\mathbf{X}(k), \dots, y(k-L) = \mathbf{W}^T(k)\mathbf{X}(k-L) \end{array} \right. \quad (7-145)$$

其中  $\boldsymbol{\varphi}(\mathbf{Y}(k)) = [\varphi_0(y(k)), \varphi_1(y(k-1)), \dots, \varphi_L(y(k-L))]^T$ ,  $\mathbf{A} = \text{diag}[\varphi_0(y(k))y(k), \varphi_1(y(k-1))y(k-1), \dots, \varphi_L(y(k-L))y(k-L)]$ 。

上面用到的各  $\varphi_i(y(k-i))$ ,  $i = 0 \sim L$ , 可以仿照式 7-80) 用下式计算:

$$\varphi_i(y(k-i)) = -\hat{p}'(y(k-i))/\hat{p}(y(k-i)), \quad i = 0 \sim L \quad (7-146)$$

$\hat{p}(\cdot)$  是假设的信号源  $S(k)$  的 pdf。如果只知道它是次高斯的 可选  $\varphi_i(y(k-i)) = y^3(k-i)$ ; 如果是超高斯的, 则选  $\varphi_i(y(k-i)) = \tanh(\gamma y(k-i))$ ,  $\gamma > 2$  或  $y(k-i) \vee |y(k-i)|$ 。详细讨论见文献 [36], [8]。

文献 [37] 还给出了一种称为 FR(filtered-regressor) 的算法, 可以用来解决复信号的均衡问题 (即上面讨论的  $S(k), x(k), y(k)$  诸时间序列皆为复序列且  $\mathbf{W}(k)$  亦为复向量) 所取的迭代计算公式是

$$\mathbf{W}(k+1) = \mathbf{W}(k) - \alpha(k)\varphi(y(k-L))\mathbf{U}^*(k) \quad (7-147)$$

其中  $\mathbf{W}(k) = [w_0(k), w_1(k), \dots, w_L(k)]^T$ ,  $y(k) = \sum w_p(k)x(k-p)$ ,  $\mathbf{U}(k) =$

$[u(k), u(k-1), \dots, u(k-L)]^T$ ,  $u(k) = \sum w_{L-p}^*(k)y(k-p)$ 。符号  $*$  表示取复共轭值。

$\varphi(\cdot)$  是取决于源信号统计特性的函数。如果源信号是次高斯的 则  $\varphi(y(k-L)) = \|y(k-L)\|^2 y(k-L)$  如果是超高斯的 则  $\varphi(y(k-L)) = \tanh(\gamma y(k-L))$ ,  $\gamma > 2$  或  $y(k-L) / \|y(k-L)\|$ 。

#### 7.9.4 时域盲解卷

这是一个未解决的研究课题, 计算与推导也相当繁杂。这里只介绍文献 [38] 所给的一种算法 (也可见文献 [8]) 不做详细推导。

此算法为随机梯度算法, 所以迭代节拍  $k$  与各时间序列的时间变量  $t$  合并且皆用  $k$  表

示。设源信号向量  $\mathbf{S}(k)$  和观察信号向量  $\mathbf{X}(k)$  都是  $N$  维复行向量，二者之间关系由式 (7-137) 描述 (将式中变元  $t$  换成  $k$ )。变换向量  $\mathbf{Y}(k)$  也是  $N$  维复行向量，由下列公式从  $\mathbf{X}(k)$  求得：

$$\mathbf{Y}(k) = \sum_{p=0}^L \mathbf{W}_p(k) \mathbf{X}(k-p) \quad (7-148)$$

$\mathbf{W}_p(k), k=0 \sim L$  是  $N \times N$  维复矩阵。算法的目标是求得一组最佳  $\mathbf{W}_p(k)$  以同时实现分离和解卷。此式中的  $L$  取决于混合-卷积系统冲激响应的延伸区间的长度 (见式 (7-136))。其迭代计算公式如下：

$$\mathbf{W}_p(k+1) = \mathbf{W}_p(k) + \alpha(k) [\mathbf{A}(k) \delta_p \mathbf{W}_p(k) - \boldsymbol{\varphi}(\mathbf{Y}(k-L)) \mathbf{U}^H(k-p)], \quad p=0 \sim L \quad (7-149)$$

式中  $\mathbf{A}(k) = \mathbf{I}$  或  $\text{diag}[\boldsymbol{\varphi}(\mathbf{Y}(k-L)) \mathbf{U}^H(k-p)], \delta_p = 0$  或  $1, \boldsymbol{\varphi}(\mathbf{Y}(k-L)) = [\varphi_1(y_1(k-L)), \varphi_2(y_2(k-L)), \dots, \varphi_N(y_N(k-L))]^T, \mathbf{U}(k-p) = [u_1(k-p), u_2(k-p), \dots, u_N(k-p)]^T$  可由下式算出：

$$\mathbf{U}(k-p) = \sum_{q=0}^L \mathbf{W}_q^H(k) \mathbf{Y}(k-p-q) \quad (7-150)$$

上列各式中的上标  $H$  表示 Hermitian 即共轭转置。函数  $\varphi_i(\cdot)$  取决于  $S_i(k)$  的统计特性，如其为次高斯，则  $\varphi_i(y_i) = \|y_i\|^2 y_i$  或  $y_i - \tanh(y_i)$ ；如其为超高斯，则  $\varphi_i(y_i) = y_i / \|y_i\|$  或  $\tanh(\gamma y_i), \gamma > 2$  或  $y_i + \tanh(y_i)$ 。

关于盲解卷的其他研究方法尚可参阅文献 [17]。下面 7.10 节介绍的 DCA 算法则从时间-空间-频率生成模型的观点出发解决盲解卷问题。

## 7.10 DCA 算法

ICA 算法解决 BSS 问题的第一条基本假设条件，即任何时刻  $t$  的源信号向量  $\mathbf{S}(t)$  各分量  $S_i(t)$  之间统计独立 (见 7.1.3 小节)。如果  $S_i(t)$  的下标  $i$  称为空间维， $t$  称为时间维，那么上述的这条假设可以称为  $\mathbf{S}(t)$  在空间上是“白色”的。至于  $\mathbf{S}(t)$  在时间上是否白色的则在 ICA 中未作任何规定。实际上对于任何  $i, S_i(t)$  在两个不同时间  $t_1$  和  $t_2$  的取值既可能统计独立 (即白色的)，也可能非统计独立 (即非白色的)。对于后者，各不同时间之间的相关性在解决 BSS 问题时还可以利用，而 ICA 并未加以利用。另外 ICA 还设观察向量  $\mathbf{X}(t)$  与源向量之间的关系是  $\mathbf{X}(t) = \mathbf{A} \mathbf{S}(t)$  (见式 (7-1))；而在盲解卷问题中  $\mathbf{X}(t) = \sum_p \mathbf{A}_p \mathbf{S}(t-p)$  (见式 (7-3))。对于前者， $\mathbf{S}(t)$  和  $\mathbf{X}(t)$  之间是单时刻对单时刻之间的映射关系；对于后者，则是多时刻对单时刻之间的对应关系。可以看到，如能将 ICA 算法中单时间点-空间至单时间点-空间映射扩展为多时间点-空间 (简称时-空阵) 至多时间点-空间映射则可以更有效地利用  $\mathbf{S}(t)$  的时间统计信息以及更有效地解决盲解卷问题。利用 DFT 还可以将时-空阵之间的映射转换为频-空阵之间的映射。按时-空阵或频-空阵的观点构成一个概率生成模型是 B. S. Everitt 于 1984 年提出的 [39]。文献 [38] 据此模型构成了用于 BSS 和盲解卷的算法。这一节将介绍这些算法的基本框架、原理和公式。由于这一算法不仅利用信号的空间信息结构 ( $i$  维) 而且利用其时间信息结构 ( $t$  维) 或者相应的

频域信息结构，且由于采用了 ICA 的计算思路，所以称为动态分量分析（dynamic component analysis 的缩写 DCA）算法。

### 7.10.1 用于 BSS 的时-空和频-空概率生成模型

在 BSS 中， $\mathbf{X}(t)$  是由单个  $\mathbf{S}(t)$  与  $\mathbf{A}$  相乘后产生的（见式 7-1）这称为单时或“瞬时”混合关系。DCA 保持 ICA 关于  $\mathbf{S}(t)$  各分量  $S_i(t)$  统计独立假设外，还假设  $S_i(t)$  是由一个白色序列  $u_i(t)$  激励一个冲激响应为  $h_i(t)$  的滤波器所产生的即  $S_i(t) = u_i(t) * h_i(t) = \sum_{\tau} u_i(\tau) h_i(t - \tau)$  符号“ $*$ ”表示卷积。下面首先按此假设给出解决 BSS 问题的 DCA 框架，然后导出此框架下的时-空和频-空概率生成模型。

#### 1. 时-空域 DCA 框架

设空间维和时间维的取值范围分别是  $i = 1 \sim N$  和  $t = 0 \sim T-1$ 。设  $t$  时刻的源信号向量是  $\mathbf{S}(t)$ ， $t = 0 \sim T-1$  诸时刻的  $\mathbf{S}(t)$  构成一个源信号时-空阵  $\mathbf{S}^t$ 。这可以表示为

$$\mathbf{S}^t = \{\mathbf{S}(0), \mathbf{S}(1), \dots, \mathbf{S}(t), \dots, \mathbf{S}(T-1)\}$$

$$\mathbf{S}(t) = [S_1(t), S_2(t), \dots, S_N(t)]^T \quad t = 0 \sim T-1$$

注意，上式右侧右上角的符号  $T$  表示转置。类似地，观察信号向量  $\mathbf{X}(t)$  和时-空阵  $\mathbf{X}^t$  可表示为

$$\mathbf{X}^t = \{\mathbf{X}(0), \mathbf{X}(1), \dots, \mathbf{X}(t), \dots, \mathbf{X}(T-1)\}$$

$$\mathbf{X}(t) = [x_1(t), x_2(t), \dots, x_N(t)]^T \quad t = 0 \sim T-1$$

注意  $\mathbf{X}(t) = \mathbf{A}\mathbf{S}(t)$ ， $t = 0 \sim T-1$ ， $\mathbf{A}$  是确定但未知的。设  $\mathbf{Y}(t) = \mathbf{W}\mathbf{X}(t)$  如  $\mathbf{W}$  选得恰当， $\mathbf{Y}(t)$  将逼近于  $\mathbf{S}(t)$  所以  $\mathbf{Y}(t)$  是  $\mathbf{S}(t)$  的估值，可以表示为  $\hat{\mathbf{S}}(t) = \mathbf{W}\mathbf{X}(t)$ 。  $t = 0 \sim T-1$  诸时刻的  $\mathbf{S}(t)$  构成如下一个源信号估值的时-空阵  $\hat{\mathbf{S}}^t$ ：

$$\hat{\mathbf{S}}^t = \{\hat{\mathbf{S}}(0), \hat{\mathbf{S}}(1), \dots, \hat{\mathbf{S}}(t), \dots, \hat{\mathbf{S}}(T-1)\}$$

$$\hat{\mathbf{S}}(t) = [\hat{S}_1(t), \hat{S}_2(t), \dots, \hat{S}_N(t)]^T, \quad t = 0 \sim T-1$$

$$\hat{\mathbf{S}}(t) = \mathbf{W}\mathbf{X}(t), \quad t = 0 \sim T-1$$

上式中各  $\hat{S}_i(t)$  是  $S_i(t)$  的估值。设  $g_i(t)$  是一个“白化”滤波器的冲激响应， $\hat{u}_i(t)$  是  $S_i(t)$  和  $g_i(t)$  的卷积，即

$$\hat{u}_i(t) = \hat{S}_i(t) * g_i(t) = \sum_{\tau=0}^{T-1} \hat{S}_i(\tau) g_i(t - \tau), \quad t = 0 \sim T-1, i = 1 \sim N \quad (7-151)$$

则  $\hat{u}_i(t)$  具有“白色”性质，它是产生  $S_i(t)$  的原白色序列的估值。 $\hat{u}_i(t)$ ， $t = 0 \sim T-1$  构成一个  $T$  维列向量  $\hat{\mathbf{U}}_i$ ， $i = 1 \sim N$  诸  $\hat{\mathbf{U}}_i$  构成时-空阵  $\hat{\mathbf{U}}^t$ ，即

$$\hat{\mathbf{U}}^t = \{\hat{\mathbf{U}}_1, \hat{\mathbf{U}}_2, \dots, \hat{\mathbf{U}}_N\}$$

$$\hat{\mathbf{U}}_i = [\hat{u}_i(0), \hat{u}_i(1), \dots, \hat{u}_i(T-1)]^T, \quad i = 1 \sim N$$

设

$$\hat{\mathbf{S}}_i = [\hat{S}_i(0), \hat{S}_i(1), \dots, \hat{S}_i(T-1)]^T, \quad i = 1 \sim N$$

则式 7-151) 可以写成下列矩阵形式

$$\hat{U}_i = G_i^t \hat{S}_i, G_i^t = \begin{bmatrix} g_i(0) & 0 & \cdots & 0 \\ g_i(1) & g_i(0) & \cdots & 0 \\ & \cdots & \cdots & \\ & \cdots & \cdots & \\ g_i(T-1) & g_i(T-2) & \cdots & g_i(0) \end{bmatrix}, i = 1 \sim N \quad (7-152)$$

导出此式时 设白化滤波器是因果的 即当  $t < 0$  时  $g_i(t) = 0$ 。这样 时-空域 DCA 框架可用下列流图描述 (见图 7-5)。图中上半所示是由  $U^t$  产生  $S^t$  及由  $S^t$  产生  $X^t$  的过程 (此过程不可见), 下半所示是由  $X^t$  产生估值  $\hat{S}^t$  再由  $\hat{S}^t$  产生估值  $\hat{U}^t$  的过程。这样, 对于任何一个观察信号的时-空阵  $X^t$  对于设定的  $W$  和各  $G_i$  可求出对应的  $\hat{S}^t$  和  $\hat{U}^t$ 。 $\hat{U}^t$  中的各  $\hat{u}_i(t)$  无论在空间维和时间维都是统计独立的, 设其 pdf 参数的总和为  $\xi^t$  则  $U^t$  的 pdf 是诸  $\hat{u}_i(t)$  和  $\xi^t$  的函数,  $\hat{S}^t$  的 pdf 则是各  $G_i^t$  和  $U^t$  的 pdf 的函数,  $X^t$  的 pdf 是  $W$  和  $\hat{S}^t$  的 pdf 的函数。最终,  $X^t$  的 pdf 是  $W$  诸  $G_i^t$  和  $\xi^t$  的函数 此 pdf 是根据图 7-5 模型计算出来的, 所以称为时-空域模型生成概率并记为  $\hat{p}_X(X^t, W, \{G_i^t, i = 1 \sim N\}, \xi^t)$  其右上标  $t$  表示时域。设  $X^t$  的真实 pdf 为  $p_X^t(X^t)$ ,  $\hat{p}_X^t(\cdot)$  和  $p_X^t(\cdot)$  之间的 Kullback-Leibler (KL) 距离可定义为目标函数 (见 7.10.2 小节)。时-空域 DCA 算法通过迭代计算求得最佳的  $W, \{G_i^t, i = 1 \sim N\}$  和  $\xi^t$  使目标函数达到极小, 从而得到最佳的  $\hat{S}^t$  即最佳的  $S^t$  估值。

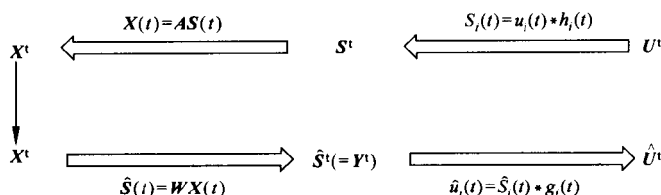


图 7-5 时-空域 DCA 的流图框架

## 2. 频率-空间域 DCA 框架

在频率-空间域中可以构成与  $X^t, S^t, \hat{U}^t$  并行的频-空阵  $X^f, \hat{S}^f, \hat{U}^f$ 。其定义列如下：

$$X^f = \{X^f(0), X^f(1), \dots, X^f(l), \dots, X^f(T-1)\}$$

$$X^f(l) = [X_1^f(l), X_2^f(l), \dots, X_N^f(l)]^T, l = 0 \sim T-1$$

$X_1^f(l)$  与  $x_i(t)$  之间关系由 DFT 和 IDFT 决定 ( $X^f$  等的右上角符号  $f$  表示频域)：

$$X_1^f(l) = \sum_{t=0}^{T-1} x_i(t) e^{-j\frac{2\pi}{T}lt}, l = 0 \sim T-1$$

$$x_i(t) = \frac{1}{T} \sum_{l=0}^{T-1} X_1^f(l) e^{j\frac{2\pi}{T}lt}, t = 0 \sim T-1$$

$$\hat{S}^f = \{\hat{S}^f(0), \hat{S}^f(1), \dots, \hat{S}^f(l), \dots, \hat{S}^f(T-1)\}$$

$$\hat{S}^f(l) = [\hat{S}_1^f(l), \hat{S}_2^f(l), \dots, \hat{S}_N^f(l)]^T, l = 0 \sim T-1$$

$$\hat{S}_i^f(l) = \sum_{t=0}^{T-1} \hat{S}_i(t) e^{-j\frac{2\pi}{T}lt}, l = 0 \sim T-1$$

$$\hat{S}_i(t) = \frac{1}{T} \sum_{l=0}^{T-1} \hat{S}_i^t(l) e^{j\frac{2\pi}{T}lt}, \quad t = 0 \sim T-1$$

注意，下列二式是并行的，即

$$\hat{\mathbf{S}}^t(l) = \mathbf{W}\mathbf{X}^t(l), \quad l = 0 \sim T-1$$

$$\hat{\mathbf{S}}(t) = \mathbf{W}\mathbf{X}(t), \quad t = 0 \sim T-1$$

最后，

$$\hat{\mathbf{U}}^t = \{\hat{\mathbf{U}}^t(0), \hat{\mathbf{U}}^t(1), \dots, \hat{\mathbf{U}}^t(l), \dots, \hat{\mathbf{U}}^t(T-1)\}$$

$$\hat{\mathbf{U}}^t(l) = \{\hat{U}_1^t(l), \hat{U}_2^t(l), \dots, \hat{U}_N^t(l)\}^T, \quad l = 0 \sim T-1$$

$$\hat{U}_i^t(l) = \sum_{t=0}^{T-1} \hat{u}_i(t) e^{-j\frac{2\pi}{T}lt}, \quad l = 0 \sim T-1$$

$$\hat{u}_i(t) = \frac{1}{T} \sum_{l=0}^{T-1} \hat{U}_i^t(l) e^{j\frac{2\pi}{T}lt}, \quad t = 0 \sim T-1$$

对式 (7-151) 左右侧施行 DFT 得到

$$\hat{U}_i^t(l) = \hat{S}_i^t(l) G_i^t(l), \quad l = 0 \sim T-1, \quad i = 1 \sim N \quad (7-153)$$

$$G_i^t(l) = \sum_{t=0}^{T-1} g_i(t) e^{-j\frac{2\pi}{T}lt}, \quad l = 0 \sim T-1$$

$$g_i(t) = \frac{1}{T} \sum_{l=0}^{T-1} G_i^t(l) e^{j\frac{2\pi}{T}lt}, \quad t = 0 \sim T-1$$

这样得到

$$\mathbf{U}^t(l) = \mathbf{G}^t(l) \mathbf{S}^t(l), \quad l = 0 \sim T-1$$

$$\mathbf{G}^t(l) = \text{diag}[G_1^t(l), G_2^t(l), \dots, G_N^t(l)] \quad (7-154)$$

在频-空域中由  $\mathbf{X}^t$  产生  $\hat{\mathbf{S}}^t$  再产生  $\hat{\mathbf{U}}^t$  的流图如图 7-6 所示。与时-空域情况类似，如果  $\hat{\mathbf{U}}^t$  的 pdf 是  $\hat{U}_i^t(l)$  和  $\xi^t$  的函数 ( $\xi^t$  是诸  $\hat{U}_i^t(l)$  的 pdf 参数的总和)，则频-空域模型生成概率可记为  $\hat{p}_{\mathbf{X}}^t(\mathbf{X}^t, \mathbf{W}, \{\mathbf{G}^t(l), l = 0 \sim T-1\}, \xi^t)$ ，设  $\mathbf{X}^t$  的真实 pdf 为  $\hat{p}_{\mathbf{X}}^t(\mathbf{X}^t)$  那么目标函数可设置为二者的 KL 距离 (见 7.10.2 小节)。

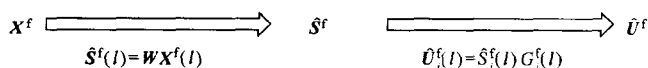


图 7-6 频-空域 DCA 流图

### 3. 时-空域和频-空域的模型生成概率计算

首先讨论时-空域情况。令各  $\hat{u}_i(t)$  的 pdf 为 MOG (mixture of Gauss) 函数表示如下 (各符号右上标  $t$  表示时域)：

$$\hat{p}_i^t(\hat{u}_i(t), \xi_i^t) = \sum_{q=1}^{Q_i^t} \frac{w_{iq}^t}{\sqrt{2\pi} \sigma_{iq}^t} \exp \left\{ -\frac{1}{2} \left( \frac{\hat{u}_i(t) - \mu_{iq}^t}{\sigma_{iq}^t} \right)^2 \right\}, \quad \forall t, i = 1 \sim N \quad (7-155)$$

其中  $\xi_i^t$  是所有  $w_{iq}, \sigma_{iq}$  和  $\mu_{iq}^t$  的总和  $Q_i^t > 1$ 。由于  $t$  取不同值时，相应的  $\hat{u}_i(t)$  统计独立 所以  $\mathbf{U}_i$  的 pdf 可用下式计算：



$$\hat{p}_i(\hat{U}_i, \xi_i^t) = \prod_{t=0}^{T-1} \hat{p}_i(\hat{u}_i(t), \xi_i^t), \quad i = 1 \sim N \quad (7-156)$$

根据式 (7-152) 和式 (7-23) 各  $S_i$  的 pdf 可用下式计算:

$$\hat{p}_i(\hat{S}_i, G_i^t, \xi_i^t) = \left| \det G_i^t \right| \hat{p}_i(\hat{U}_i, \xi_i^t) \Big|_{\hat{U}_i = G_i^t \hat{S}_i}, \quad i = 1 \sim N \quad (7-157)$$

其中  $\left| \det G_i^t \right| = [g_i(0)]^T$ 。注意到  $i$  取不同值时相应的诸  $S_i$  统计独立  $\hat{S}^t = \{\hat{S}_1, \hat{S}_2, \dots, \hat{S}_N\}$  则  $S^t$  的 pdf 可用下式计算:

$$\hat{p}(\hat{S}^t, G^t, \xi^t) = \prod_{i=1}^N \hat{p}_i(\hat{S}_i, G_i^t, \xi_i^t) \quad (7-158)$$

其中  $G^t$  是各  $G_i^t, i = 1 \sim N$  的总和,  $\xi^t$  是各  $\xi_i^t$  的总和。注意  $\hat{S}^t$  还可表示为  $\{\hat{S}^t(0), \hat{S}^t(1), \dots, \hat{S}^t(T-1)\}$  且  $S(t) = WX(t), t = 0 \sim T-1$ 。再次利用式 (7-23) 可以导出  $X^t = \{X(0), X(1), \dots, X(T-1)\}$  的模型生成 pdf 为<sup>①</sup>

$$\begin{aligned} \hat{p}(X^t, W, G^t, \xi^t) &= \left| \det W \right|^T \hat{p}(\hat{S}^t, G^t, \xi^t) \\ &= \left| \det W \right|^T \prod_{i=1}^N [g_i(0)]^T \prod_{i=1}^N \prod_{t=0}^{T-1} \hat{p}_i(\hat{u}_i(t), \xi_i^t) \Bigg|_{\substack{\hat{u}_i(t) = \hat{S}_i^t(t) \cdot g_i(t) \\ \hat{S}^t(t) = WX(t)}} \quad (7-159) \end{aligned}$$

其次给出频-空域的情况。与时-空域情况的区别在于  $\hat{U}_i^t(l)$  是一个复数且  $\hat{U}_i^t(l) = [\hat{U}_i^t(T-1-l)]^*, l = 0 \sim T-1, *$  表示取复共轭值。若  $\hat{U}_i^t(l)$  的实部和虚部分别表示为  $\hat{U}_i^R(l)$  和  $\hat{U}_i^I(l)$  则仿照时-空域的推导方法 (考虑到  $\hat{U}_i^t(l)$  的上述性质且  $G_i^t(l), X_i^t(l)$  也具有类似性, 在推导上需采取一些技巧, 见文献[39]) 可以求得  $X^t = \{X^t(0), X^t(1), \dots, X^t(T-1)\}$  的模型生成 pdf 为

$$\hat{p}(X^t, W, G^t(l), \xi^t) = \left| \det W \right|^T \prod_{i=1}^N \prod_{l=0}^{T-1} \left| G_i^t(l) \right| \prod_{i=1}^N \prod_{l=0}^{\frac{T}{2}-1} \hat{p}_i^t(\hat{U}_i^t(l), \xi_i^t) \Bigg|_{\substack{\hat{U}_i^t(l) = \hat{S}_i^t(l) \cdot G_i^t(l) \\ \hat{S}^t(l) = WX^t(l)}} \quad (7-160)$$

其中  $|G_i^t(l)|$  为  $G_i^t(l)$  的模。 $\hat{p}_i^t(\cdot)$  由下式计算:

$$\hat{p}_i^t(\hat{U}_i^t(l), \xi_i^t) = \begin{cases} \hat{p}_i^t(\hat{U}_i^R(l), \xi_i^t) \cdot \hat{p}_i^t(\hat{U}_i^I(l), \xi_i^t), & l = 1 \sim \frac{T}{2} - 1 \\ \hat{p}_i^t(\hat{U}_i^R(l), \xi_i^t), & l = 0, \quad \frac{T}{2} \end{cases}$$

设  $V$  表示  $\hat{U}_i^R(l)$  或  $\hat{U}_i^I(l)$  则  $\hat{p}_i^t(\hat{V}_i, \xi_i^t)$  可按下式计算:

$$\hat{p}_i^t(\hat{V}_i, \xi_i^t) = \sum_{r=1}^{R_i^t} \frac{w_{ir}^t}{\sqrt{2\pi}\sigma_{ir}^t} \exp \left\{ -\frac{1}{2} \left( \frac{\hat{V}_i - \mu_{ir}^t}{\sigma_{ir}^t} \right)^2 \right\}, \quad i = 1 \sim N \quad (7-161)$$

其中  $\xi_i^t$  是  $w_{ir}^t, \mu_{ir}^t$  和  $\sigma_{ir}^t$  的总和,  $R_i^t > 1$ 。

① 式 (7-159) 中取  $|\det W|$  的  $T$  次幂是在各  $\hat{S}^t(t)$  当  $t$  取不同值时统计独立的情况下才成立。若存在相关性, 则应取小于  $T$  的幂值; 例如, 在完全相关的情况下, 应取 1 次幂。

#### 4. 时、频混合空域的模型生成概率计算

由于  $\mathbf{X}^i$  和  $\mathbf{X}^i$  之间、 $\hat{\mathbf{S}}^i$  和  $\hat{\mathbf{S}}^i$  之间、 $\hat{\mathbf{U}}^i$  和  $\hat{\mathbf{U}}^i$  之间可通过 DFT 和 IDFT 相互转换 所以模型生成 pdf 可以采用时、频混合的方式计算。下面给出在实际运算中效率最高的一种：

$$\hat{p}(\mathbf{X}^i, \mathbf{W}, \mathbf{G}^i(l), \xi^i) = \left| \det \mathbf{W} \right|^T \prod_{i=1}^N \prod_{l=0}^{T-1} \left| G_i^i(l) \right| \prod_{i=1}^N \prod_{l=0}^{T-1} \hat{p}_i^i(\hat{u}_i(t), \xi_i^i) \quad \left\{ \begin{array}{l} \hat{u}_i(t) = \text{IDFT}(\hat{U}_i^i(l)) \\ \hat{U}_i^i(l) = \hat{\mathbf{S}}_i^i(l) \cdot \mathbf{G}_i^i(l) \\ \hat{\mathbf{S}}_i^i(l) = \mathbf{W} \mathbf{X}_i^i(l) \\ \mathbf{X}_i^i(l) = \text{DFT}(x_i(t)) \end{array} \right. \quad (7-162)$$

#### 7.10.2 用于 BSS 的 DCA 学习算法

为求得最佳的参数  $\mathbf{W}, \mathbf{G}^i(l)$  (或  $g_i(t)$ ),  $\xi^i$  (或  $\xi_i$ ) 首先应确立目标函数来比较模型产生 pdf  $\hat{p}(\cdot)$  和实际 pdf  $p(\cdot)$  的相似度。与 7.3.3 小节给出的 ICA 目标函数相同, DCA 也取  $p(\mathbf{X}^i)$  和  $\hat{p}(\mathbf{X}^i)$  之间 (或  $p(\mathbf{X}^i)$  和  $\hat{p}(\mathbf{X}^i)$  之间) 的 Kullback-Leibler 距离作为目标函数 (见式 7-30)。如果采用时频混合方式计算  $\hat{p}(\mathbf{X}^i)$  (或 7-162) 则目标函数是  $\mathbf{W}, \mathbf{G}^i(l), \xi^i$  的函数, 可表示如下:

$$\begin{aligned} L_{\text{DCA-IFT}}(\mathbf{W}, \mathbf{G}^i(l), \xi^i) &= \text{KL}(p(\mathbf{X}^i) \parallel \hat{p}(\mathbf{X}^i, \mathbf{W}, \mathbf{G}^i(l), \xi^i)) \\ &= \int_{\mathbf{X}^i} p(\mathbf{X}^i) \ln \left[ \frac{p(\mathbf{X}^i)}{\hat{p}(\mathbf{X}^i, \mathbf{W}, \mathbf{G}^i(l), \xi^i)} \right] d\mathbf{X}^i \quad (7-163) \end{aligned}$$

下标 DCA-IFT 表示用时频混合方式的 DCA 解 BSS 算法。由于  $\int_{\mathbf{X}^i} p(\mathbf{X}^i) \ln[p(\mathbf{X}^i)] d\mathbf{X}^i$  是与  $\mathbf{W}$  等参数无关的项, 故可从目标函数中剔除。此外, 式 7-163 的全集合平均不可能实现, 只能用批平均的近似目标函数替代 (参见式 (7-31))。设有  $M$  个时空阵  $\mathbf{X}_m^i = \{\mathbf{X}(mT), \mathbf{X}(mT+1), \dots, \mathbf{X}(mT+T-1)\}, m=0 \sim M-1$ , 则近似目标函数定义为

$$\tilde{L}_{\text{DCA-IFT}}(\mathbf{W}, \mathbf{G}^i(l), \xi^i) = \frac{1}{M} \sum_{m=0}^{M-1} -\frac{1}{T} \ln[\hat{p}(\mathbf{X}_m^i, \mathbf{W}, \mathbf{G}^i(l), \xi^i)] \quad (7-164)$$

将式(7-162)代入式(7-164), 且用  $\langle f(\cdot) \rangle$  表示  $\frac{1}{M} \sum_{m=1}^M f(\cdot)$ , 则式(7-164) 可表示为

$$\tilde{L}_{\text{DCA-IFT}}(\cdot) = -\ln|\det \mathbf{W}| - \frac{1}{T} \sum_{i=1}^N \sum_{l=0}^{T-1} \ln|G_i^i(l)| - \frac{1}{T} \sum_{i=1}^N \sum_{l=0}^{T-1} \langle \ln \hat{p}_i^i(\hat{u}_i(t), \xi_i^i) \rangle \quad (7-165)$$

为简化 式 7-162 所附缀的由  $x_i(t)$  求  $\hat{u}_i(t)$  诸式在此式中略而不书。

现在可以和 ICA 算法相似, 采用相对梯度计算思路求最佳参数  $\mathbf{W}, \mathbf{G}^i(l)$  (或对应的  $g_i(t)$ ),  $\xi^i$  使  $L_{\text{DCA-IFT}}$  达到最小。对于  $\mathbf{W}$ , 可以按式 7-56) 进行迭代计算 (迭代节拍为  $k$ ) 即

$$\Delta \mathbf{W}(k) = -\alpha(k) \nabla_{\mathbf{W}} \tilde{L}_{\text{DCA-IFT}}(\cdot) \mathbf{W}^T \mathbf{W} \Big|_{\mathbf{W}=\mathbf{W}(k)}$$

按照文献[39] 的推导 得到

$$\Delta \mathbf{W}(k) = \alpha(k) \left[ \mathbf{I} - \left\langle \sum_{t=0}^{T-1} \mathbf{G}^{(k)}(t) (\Psi^{(k)}(t) * (\mathbf{S}^{(k)}(-t))^T) \right\rangle \right] \mathbf{W}(k) \quad (7-166)$$

其中  $\alpha(k) > 0$  是随  $k$  而变化的步幅,  $\mathbf{I}$  是  $N \times N$  维单位阵,  $\Psi^{(k)}(t)$  定义如下:

$$\Psi^{(k)}(t) = [\psi_1^{(k)}(t), \psi_2^{(k)}(t), \dots, \psi_N^{(k)}(t)]^T, \quad t = 0 \sim T-1$$

$$\phi_i^{(k)}(t) = -\frac{1}{T} \frac{\partial \ln \hat{p}_i^t(\hat{u}_i(t), \xi_i^t)}{\partial \hat{u}_i(t)} \bigg|_{\substack{\hat{u}_i(t) = \hat{u}_i^{(k)}(t), \\ \xi_i^t = \xi_i^t(k)}} \quad i = 1 \sim N \quad (7-167)$$

各  $\hat{u}_i^{(k)}(t)$  是根据参数  $\mathbf{W}(k), \mathbf{G}^{(k)}(t)$  由  $\mathbf{X}^t$  求得的,  $\xi_i^t(k)$  是节拍  $k$  的  $\hat{u}_i(t)$  的 pdf 参数。 $\mathbf{S}^{(k)}(t)$  已在 7.10.1 小节中定义,  $\mathbf{S}^{(k)}(t) = [\hat{S}_1^{(k)}(t), \hat{S}_2^{(k)}(t), \dots, \hat{S}_N^{(k)}(t)]^T$  各  $\hat{S}_i^{(k)}(t)$  是按  $\mathbf{W}(k)$  由  $\mathbf{X}(t)$  求得的。 $(\boldsymbol{\psi}^{(k)}(t) * (\hat{\mathbf{S}}^{(k)}(-t))^T)$  是一个  $N \times N$  维矩阵, 它的第  $i$  行第  $j$  列元素为  $\phi_i^{(k)}(t) * \hat{S}_j^{(k)}(-t)$ , 可由下式求得:

$$\phi_i^{(k)}(t) * \hat{S}_j^{(k)}(-t) = \sum_{\tau=0}^{T-1} \phi_i^{(k)}(\tau) \hat{S}_j^{(k)}(\tau-t) \quad (7-168)$$

$\mathbf{G}^{(k)}(t), t = 0 \sim T-1$  是  $T$  个定义如下的  $N \times N$  维对角阵:

$$\mathbf{G}^{(k)}(t) = \text{diag}[g_1^{(k)}(t), g_2^{(k)}(t), \dots, g_N^{(k)}(t)] \quad (7-169)$$

$g_i^{(k)}(t)$  是节拍  $k$  的白化滤波器响应。

对于白化滤波器响应  $g_i^{(k)}(t), i = 1 \sim N, t = 0 \sim T-1$  其迭代计算公式是 见文献 [39])

$$g_i^{(k+1)}(t) = g_i^{(k)}(t) + \Delta g_i^{(k)}(t) \quad (7-170)$$

$$\Delta g_i^{(k)}(t) = \alpha(k) \left[ g_i^{(k)}(t) - \sum_{\tau=0}^{T-1} f_i^{(k)}(\tau) g_i^{(k)}(\tau+t) \right]$$

$$f_i^{(k)}(\tau) = \sum_{m=0}^{T-\tau-1} \phi_i^{(k)}(m) \hat{u}_i^{(k)}(m+\tau) \quad (7-171)$$

为求得各  $\xi_i^t$  的最佳值 首先  $\hat{p}_i^t(\hat{u}_i(t), \xi_i^t)$  的表示式 (式 7-155) 中的各权系数  $w_{iq}^t$  应满足

$$\sum_{q=1}^{Q_i^t} w_{iq}^t = 1 \quad (7-172)$$

为此 用  $\gamma_{iq}^t$  来表示  $w_{iq}^t$ ,

$$w_{iq}^t = \frac{\exp(\gamma_{iq}^t)}{\sum_{\beta=1}^{Q_i^t} \exp(\gamma_{i\beta}^t)} \quad (7-173)$$

这样  $\xi_i^t$  的参数为  $\gamma_{iq}^t, \sigma_{iq}^t$  和  $\mu_{iq}^t, q = 1 \sim Q_i^t$ 。按照迭代节拍  $k$  进行迭代计算时, 各参数的调整量可按下式计算:

$$\left. \begin{aligned} \Delta \gamma_{iq}^t(k) &= -\alpha(k) \frac{\partial \ln \hat{p}_i^t(\hat{u}_i(t), \xi_i^t)}{\partial \gamma_{iq}^t} \bigg|_{\xi_i^t(k)} = \alpha(k) [f_{iq}^t(k) - w_{iq}^t(k)] \\ \Delta \mu_{iq}^t(k) &= -\alpha(k) \frac{\partial \ln \hat{p}_i^t(\hat{u}_i(t), \xi_i^t)}{\partial \mu_{iq}^t} \bigg|_{\xi_i^t(k)} = \alpha(k) \left[ \frac{\hat{u}_i^{(k)}(t) - \mu_{iq}^t(k)}{(\sigma_{iq}^t(k))^2} \right] f_{iq}^t(k) \\ \Delta \sigma_{iq}^t(k) &= -\alpha(k) \frac{\partial \ln \hat{p}_i^t(\hat{u}_i(t), \xi_i^t)}{\partial \sigma_{iq}^t} \bigg|_{\xi_i^t(k)} = \alpha(k) \left[ \frac{(\hat{u}_i^{(k)}(t) - \mu_{iq}^t(k))^2}{(\sigma_{iq}^t(k))^2} - \frac{1}{\sigma_{iq}^t(k)} \right] f_{iq}^t(k) \\ f_{iq}^t(k) &= \frac{1}{\hat{p}_i^t(\hat{u}_i^{(k)}(t), \xi_i^t(k))} \cdot \frac{w_{iq}^t(k)}{\sqrt{2\pi\sigma_{iq}^t(k)}} \exp \left\{ -\frac{1}{2} \left[ \frac{\hat{u}_i^{(k)}(t) - \mu_{iq}^t(k)}{\sigma_{iq}^t(k)} \right]^2 \right\} \end{aligned} \right\} \quad (7-174)$$

注意，上列诸式中只表示了一个时间点  $t$ ，而计算调整量时应取  $t = 0 \sim T-1$  各时间点调整量的均值。如有  $M$  个时空阵，还应对其取平均。

### 7.10.3 用于盲分离—解卷 多道盲解卷 的概率生成模型和 DCA 学习算法

为讨论方便 将 7.9.1 小节的因果混合-卷积系统计算式 (式 (7-137)) 改写为下列公式：

$$\mathbf{X}(t) = \sum_{\tau=0}^L \mathbf{A}(\tau) \mathbf{U}(t-\tau), \quad t = 0 \sim T-1 \quad (7-175)$$

这里用  $\mathbf{U}(t)$  替代  $\mathbf{S}(t)$  来表示一个空间和时间都是白色的源信号向量。设  $\mathbf{U}(t) = [u_1(t), \dots, u_N(t)]^T$  则当  $i \neq j$  时  $u_i(t)$  与  $u_j(t)$  统计独立；当  $t_1 \neq t_2$  时  $u_i(t_1)$  与  $u_i(t_2)$  统计独立。 $t$  时刻的观察信号向量是  $\mathbf{X}(t) = [x_1(t), \dots, x_N(t)]^T$ 。 $\mathbf{A}(\tau) = (a_{ij}(\tau))$ ,  $\tau = 0 \sim L$  是未知的  $N \times N$  维混合-冲激响应阵。待完成的任务是根据已知的  $\mathbf{X}(t)$ ,  $t = 0 \sim T-1$  求未知的源信号  $\mathbf{U}(t)$ ,  $t = 0 \sim T-1$ 。下面给出实现 DCA 多道盲解卷的框架。首先与 BSS 情况相同， $\mathbf{X}(t)$ ,  $t = 0 \sim T-1$  构成一个时-空阵  $\mathbf{X}^t$ ，

$$\mathbf{X}^t = \{\mathbf{X}(0), \mathbf{X}(1), \dots, \mathbf{X}(T-1)\}$$

$$\mathbf{X}(t) = [x_1(t), x_2(t), \dots, x_N(t)]^T, \quad t = 0 \sim T-1$$

设  $\mathbf{W}(t)$ ,  $t = 0 \sim T-1$  是实现分离—解卷的系列  $T$  个  $N \times N$  维变换阵，其第  $i$  行  $j$  列元素为  $w_{ij}(t)$ 。设  $\mathbf{Y}(t)$  是  $\mathbf{W}(t)$  和  $\mathbf{X}(t)$  的卷积，若  $\mathbf{W}(t)$  选得合适， $\mathbf{Y}(t)$  将逼近于  $\mathbf{U}(t)$ 。这时  $\mathbf{Y}(t)$  将成为  $\mathbf{U}(t)$  的估值，可表示为  $\hat{\mathbf{U}}(t)$  即有

$$\mathbf{U}(t) = \mathbf{W}(t) * \mathbf{X}(t) = \sum_{\tau=0}^{T-1} \mathbf{W}(\tau) \mathbf{X}(t-\tau), \quad t = 0 \sim T-1 \quad (7-176)$$

这时  $\mathbf{U}(t)$ ,  $t = 0 \sim T-1$ ，构成一个源信号估值时-空阵  $\hat{\mathbf{U}}^t$ 。

$$\hat{\mathbf{U}}^t = \{\mathbf{U}(0), \mathbf{U}(1), \dots, \mathbf{U}(t), \dots, \mathbf{U}(T-1)\}$$

$$\mathbf{U}(t) = [\hat{u}_1(t), \hat{u}_2(t), \dots, \hat{u}_N(t)]^T, \quad t = 0 \sim T-1$$

其中  $\hat{u}_i(t)$  是真实源信号  $u_i(t)$  的估值。实现盲分离—解卷的标准是各  $\hat{u}_i(t)$  在空间和时间两维上都是白色的。这就是盲分离—解卷的时-空域 DCA 框架。类似于 BSS 情况，其频-空域框架为将各时域变量通过 DFT 转到频域，反之，用 IDFT 则可将频域变量转回时域。 $\mathbf{X}^t$  和  $\mathbf{X}(t)$ ,  $t = 0 \sim T-1$  在频域的对项为  $\mathbf{X}^f$  和  $\mathbf{X}^f(l)$ ,  $l = 0 \sim T-1$ 。二者之间关系如 7.10.2 小节所列。 $\hat{\mathbf{U}}^t$  和  $\hat{\mathbf{U}}(t)$ ,  $t = 0 \sim T-1$  在频域的对项为  $\hat{\mathbf{U}}^f$  和  $\hat{\mathbf{U}}^f(l)$ ,  $l = 0 \sim T-1$ ，

$$\hat{\mathbf{U}}^f = \{\hat{\mathbf{U}}^f(0), \hat{\mathbf{U}}^f(1), \dots, \hat{\mathbf{U}}^f(l), \dots, \hat{\mathbf{U}}^f(T-1)\}$$

$$\hat{\mathbf{U}}^f(l) = [\hat{U}_1^f(l), \hat{U}_2^f(l), \dots, \hat{U}_N^f(l)]^T, \quad l = 0 \sim T-1$$

时频与频域之间的转换关系是

$$\begin{aligned} \hat{U}_i^f(l) &= \sum_{t=0}^{T-1} \hat{u}_i(t) e^{-j \frac{2\pi}{T} lt}, \quad l = 0 \sim T-1 \\ \hat{u}_i(t) &= \frac{1}{T} \sum_{l=0}^{T-1} \hat{U}_i^f(l) e^{j \frac{2\pi}{T} lt}, \quad t = 0 \sim T-1 \end{aligned}$$

$\hat{U}^l(l)$  和  $\mathbf{X}^l(l)$  之间关系可通过对式 (7-176) 两侧施行 DFT 来获得：

$$\mathbf{U}^l(l) = \mathbf{W}^l(l) \mathbf{X}^l(l), \quad l = 0 \sim T-1 \quad (7-177)$$

其中  $\mathbf{W}^l(l)$  是  $\mathbf{W}(t)$  的 DFT 设其第  $i$  行  $j$  列元素为  $w_{ij}^l(l)$  则其与  $w_{ij}(t)$  的关系是

$$w_{ij}^l(l) = \sum_{t=0}^{T-1} w_{ij}(t) e^{-j\frac{2\pi}{T}lt}, \quad l = 0 \sim T-1$$

$$w_{ij}(t) = \frac{1}{T} \sum_{l=0}^{T-1} w_{ij}^l(l) e^{j\frac{2\pi}{T}lt}, \quad t = 0 \sim T-1$$

与 BSS 情况相同 令各  $\hat{u}_i(t)$  的 pdf 为 MOG 函数且可用式 (7-155) 表示。这样 可以构成时-空域、频-空域和时频混合-空域的  $\mathbf{X}^l$  模型生成概率。由于时频混合情况的效果最好，所以下面只给出这种情况的  $\mathbf{X}^l$  生成概率。其推导见文献[39] 此处从略。

$$\hat{p}(\mathbf{X}^l, \mathbf{W}^l(l), \boldsymbol{\xi}^l) = \prod_{l=0}^{T-1} \left| \det \mathbf{W}^l(l) \right| \prod_{i=1}^N \prod_{t=0}^{T-1} \hat{p}_i(\hat{u}_i(t), \boldsymbol{\xi}_i^l) \left\{ \begin{array}{l} \hat{u}_i(t) = \text{IDFT}\{\hat{U}_i^l(l)\} \\ \hat{\mathbf{U}}^l(l) = \mathbf{W}^l(l) \cdot \mathbf{X}^l(l) \\ \mathbf{X}_i^l(l) = \text{DFT}\{x_i(t)\} \end{array} \right. \quad (7-178)$$

其中  $\hat{p}_i(\hat{u}_i(t), \boldsymbol{\xi}_i^l)$  由式 (7-155) 计算,  $\boldsymbol{\xi}^l$  是各  $\boldsymbol{\xi}_i^l$  的总和。接着 与 7.10.2 小节关于 DCA 的学习算法类似, 为求最佳的参数  $\mathbf{W}^l(l), l = 0 \sim T-1$  和  $\boldsymbol{\xi}^l$ , 可定义一目标函数  $L_{\text{DCA-CFT}}(\mathbf{W}^l(l), \boldsymbol{\xi}^l)$  其计算公式与式 (7-163) 相同 (只是将该式中的  $\hat{p}(\mathbf{X}^l, \mathbf{W}, \mathbf{G}^l(l), \boldsymbol{\xi}^l)$  用式 (7-178) 给出的  $\hat{p}(\mathbf{X}^l, \mathbf{W}^l(l), \boldsymbol{\xi}^l)$  替代) 同样 在实际计算中此目标函数只能用下列取批平均的近似目标函数替代：

$$\begin{aligned} \tilde{L}_{\text{DCA-CFT}}(\mathbf{W}^l(l), \boldsymbol{\xi}^l) &= \frac{1}{M} \sum_{m=1}^M -\frac{1}{T} \ln[\hat{p}(\mathbf{X}_m^l, \mathbf{W}^l(l), \boldsymbol{\xi}^l)] \\ &= -\frac{1}{T} \sum_{l=0}^{T-1} \ln|\det \mathbf{W}^l(l)| - \frac{1}{T} \sum_{i=1}^N \sum_{t=0}^{T-1} \langle \ln \hat{p}_i(\hat{u}_i(t), \boldsymbol{\xi}_i^l) \rangle \end{aligned} \quad (7-179)$$

DCA-CFT 表示用时频混合方式的 DCA 多道盲解卷算法  $\langle \cdot \rangle$  表示对  $m = 1 \sim M$  诸样本的取平均；如果  $M = 1$ ，则目标函数只由一个观察信号的时空阵来计算（见 7.10.2 小节关于  $\mathbf{X}_m^l$  及  $\langle \cdot \rangle$  的定义）

为了求得使此目标函数达到极小的最佳  $\mathbf{W}^l(l), l = 0 \sim T-1$  可以采用相对梯度思路 按节拍  $k$  进行迭代计算。迭代计算首先针对  $\mathbf{W}^l(l)$  进行 然后通过 IDFT 将其转换为针对  $\mathbf{W}(t)$  的迭代计算。 $\mathbf{W}(t)$  按节拍  $k = 0, 1, 2, \dots$  进行调整，所以可表示为  $k$  的函数  $\mathbf{W}^{(k)}(t)$ 。 $\mathbf{W}^{(k+1)}(t) = \mathbf{W}^{(k)}(t) + \Delta \mathbf{W}^{(k)}(t)$ ,  $\Delta \mathbf{W}^{(k)}(t)$  应为节拍  $k$  时  $\tilde{L}_{\text{DCA-CFT}}(\cdot)$  对于  $\mathbf{W}(t)$  的负相对梯度值乘以步幅  $\alpha(k)$ 。按照文献[39] 的推导，其计算公式如下列：

$$\left. \begin{aligned} \Delta \mathbf{W}^{(k)}(t) &= \alpha(k) \left[ \mathbf{W}^{(k)}(t) - \sum_{\tau=0}^{T-1} \langle \mathbf{F}^{(k)}(-\tau) \rangle \mathbf{W}^{(k)}(\tau+t) \right], \quad t = 0 \sim T-1 \\ \mathbf{F}^{(k)}(\tau) &= [\boldsymbol{\psi}^{(k)}(\tau) * (\hat{\mathbf{U}}^{(k)}(-\tau))^T] \\ \boldsymbol{\psi}^{(k)}(\tau) &= [\psi_1^{(k)}(\tau), \dots, \psi_N^{(k)}(\tau)]^T, \hat{\mathbf{U}}^{(k)}(\tau) = [\hat{u}_1^{(k)}(\tau), \dots, \hat{u}_N^{(k)}(\tau)]^T \\ \psi_i^{(k)}(\tau) &= -\frac{1}{T} \frac{\partial \ln \hat{p}_i(\hat{u}_i(\tau), \boldsymbol{\xi}_i^l)}{\partial \hat{u}_i(\tau)} \bigg|_{\substack{\hat{u}_i(\tau) = \hat{u}_i^{(k)}(\tau) \\ \boldsymbol{\xi}_i^l = \boldsymbol{\xi}_i^{(k)}}} \end{aligned} \right\} \quad (7-180)$$

$\mathbf{F}^{(k)}(-\tau)$  是一个  $N \times N$  维阵 其第  $i$  行  $j$  列元素  $f_{ij}^{(k)}(-\tau)$  由下式计算：

$$f_{ij}^{(k)}(-\tau) = \sum_{m=0}^{T-1} \phi_i^{(k)}(m) \hat{u}_j^{(k)}(m+\tau) \quad (7-181)$$

注意,各  $\hat{u}_i^{(k)}(\tau)$  是由  $\mathbf{X}^i$  和  $\mathbf{W}^{(k)}(t)$  计算得到的。 $\langle \cdot \rangle$  表示对  $M$  个时-空阵取平均。 $\xi_i^i$  中各参数  $\gamma_{iq}^i, \sigma_{iq}^i$  和  $\mu_{iq}^i$  的迭代计算公式与 7.10.2 小节给出的式 (7-174) 完全相同。

## 7.11 IFA 算法

用 ICA 算法解决 BSS 问题的优点是：分离效果好、计算效率高且具有等价变化性。其缺点是：必须保持观察信号数等于源信号数（即  $M = N$ ），只适用观察信号无迭加噪声或迭加噪声很小的情况 此外 在 ICA 算法中对于源信号 pdf 的确定或学习始终是一个瓶颈性难题。在统计信号处理领域中还存在另一种源信号估计算法，这就是因子分析算法（factor analysis, FA）。在标准 FA 算法中，设有  $N$  个观察不到的源信号（称为因子） $S_j, j = 1 \sim N$  它们经过线性组合 加上迭加噪声  $N_i$  后 形成  $M$  个观察信号  $x_i, i = 1 \sim M$ 。即

$$x_i = \sum_{j=1}^N a_{ij} S_j + N_i, \quad i = 1 \sim M$$

其中假设各  $S_j$  是统计独立且都具有高斯分布的随机变量，各  $N_i$  也是均值为 0 且统计独立的高斯随机变量。FA 采用一种高效的 EM 算法（详见 7.11.3 小节的介绍）来估计出各  $N$  的方差、各  $S_j$  的均值和方差以及混合阵  $a_{ij}$ 。然后用这些估计参数和各  $x_i$  对各  $S_j$  作出最大似然估计<sup>[42]</sup>。FA 算法的局限是各独立信号源只具有高斯分布特性，为此文献<sup>[41]</sup>提出了称为 IFA (Independent factor analysis) 的两种改进算法，此二法中各信号源的 pdf 皆取为 MOG 函数。有噪声的 IFA 算法可以在噪声较强且  $M \neq N$  时对各源信号作出估计。但是，当噪声较弱时，这种算法趋近于 PCA，因此分离效果较差。无噪声的 IFA 算法将 ICA 算法与 FA 算法相结合，除了可以解决  $M \neq N$  和源信号 pdf 学习的困难以外，还具有效率高的特点。下面依次介绍这两种算法的框架、参数学习、源信号恢复和有关的问题。

### 7.11.1 有噪 IFA 的框架

设有  $N$  个统计独立的源信号  $S_i(t), i = 1 \sim N$  设各源信号的 pdf 与  $t$  无关。为简化，可将时序变量  $t$  略去 将  $S_i(t)$  表示为  $S_i$ 。 $N$  个  $S_i$  构成一个随机列向量  $\mathbf{S} = [S_1, \dots, S_N]^T$ 。设每个源信号  $S_i$  的 pdf 记为  $\hat{p}_i(S_i | \boldsymbol{\Omega}_i, \boldsymbol{\xi}_i)$  它是用下列公式表示的 MOG 函数：

$$\hat{p}_i(S_i | \boldsymbol{\Omega}_i, \boldsymbol{\xi}_i) = \sum_{q_i=1}^{Q_i} \hat{p}_i(q_i) \hat{p}_i(S_i | q_i) = \sum_{q_i=1}^{Q_i} \hat{p}_i(q_i) \frac{1}{\sqrt{2\pi}\sigma_{iq_i}} \exp\left[-\frac{1}{2} \left(\frac{S_i - \mu_{iq_i}}{\sigma_{iq_i}}\right)^2\right] \quad (7-182)$$

其中  $Q_i > 1, Q_i$  为 pdf 中高斯函数的个数,  $q_i = 1 \sim Q_i$  表示各高斯函数的编号，相应的均值和方差为  $\mu_{iq_i}$  和  $\sigma_{iq_i}$ 。设  $\hat{p}_i(q_i) = \omega_{iq_i}$ ，是各高斯函数的加权系数；如果把每个高斯函数看成一种“状态”那么  $\hat{p}_i(q_i)$  就是编号为  $q_i$  的状态出现的概率。 $\boldsymbol{\xi}_i$  表示此 pdf 中各高斯函数的参数  $\mu_{iq_i}$  和  $\sigma_{iq_i}$  的总和， $\boldsymbol{\Omega}_i$  表示各  $\omega_{iq_i}$  的总和。由于  $\mathbf{S}$  的各分量  $S_i$  统计独立 其 pdf 可

用下式计算：

$$\begin{aligned}\hat{p}(S | \Omega, \xi) &= \prod_{i=1}^N \hat{p}_i(S_i | \Omega_i, \xi_i) = \prod_{i=1}^N \left\{ \sum_{q_i=1}^{Q_i} \hat{p}_i(q_i) \hat{p}_i(S_i | q_i) \right\} \\ &= \sum_{q_1=1}^{Q_1} \cdots \sum_{q_N=1}^{Q_N} \left\{ \prod_{i=1}^N \hat{p}_i(q_i) \right\} \left\{ \prod_{i=1}^N \frac{1}{\sqrt{2\pi}\sigma_{iq_i}} \right\} \exp \left\{ -\frac{1}{2} \sum_{i=1}^N \left( \frac{S_i - \mu_{iq_i}}{\sigma_{iq_i}} \right)^2 \right\} \quad (7-183)\end{aligned}$$

其中  $\Omega$  表示各  $\Omega_i$  的总和,  $\xi$  表示各  $\xi_i$  的总和。如上文所述 可将  $q_i$  的取值看成  $S_i$  所处的一种状态。这样 可以设置一个  $N$  维状态向量  $q, q = [q_1, \cdots, q_N]^T$  共有  $\prod Q$  种不同的状态向量 (为简化起见 下文将向量二字略去) 每一种状态  $q$  的出现概率记为  $\hat{p}(q | \Omega)$  用下式计算：

$$\hat{p}(q | \Omega) = \prod_{i=1}^N \hat{p}_i(q_i) = \prod_{i=1}^N \omega_{iq_i} \quad (7-184)$$

在  $q$  状态下  $S$  的出现概率记为  $\hat{p}(S | q, \xi)$ ,

$$\hat{p}(S | q, \xi) = \prod_{i=1}^N \hat{p}_i(S_i | q_i) = \left\{ \prod_{i=1}^N \frac{1}{\sqrt{2\pi}\sigma_{iq_i}} \right\} \exp \left\{ -\frac{1}{2} \sum_{i=1}^N \left( \frac{S_i - \mu_{iq_i}}{\sigma_{iq_i}} \right)^2 \right\} \quad (7-185)$$

这样, 式 7-183) 可以重写如下：

$$\hat{p}(S | \Omega, \xi) = \sum_q \hat{p}(q, S | \Omega, \xi) = \sum_q \hat{p}(q | \Omega) \hat{p}(S | q, \xi) \quad (7-186)$$

其中  $\sum_q$  表示  $\sum_{q_1} \cdots \sum_{q_N}$ 。如果设  $\mu_q = [\mu_{1q_1}, \cdots, \mu_{Nq_N}]^T, V_q = \text{diag}[\sigma_{1q_1}^2, \cdots, \sigma_{Nq_N}^2]$ , 则式 (7-185) 和式 7-186) 中的  $\hat{p}(S | q, \xi)$  可表示为  $\hat{p}(S | q, \xi_q)$ ,

$$\hat{p}(S | q, \xi_q) = (2\pi)^{-\frac{N}{2}} (\det V_q)^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (S - \mu_q)^T V_q^{-1} (S - \mu_q) \right\} \quad (7-187)$$

其中  $\xi_q$  包含  $\mu_q$  和  $V_q$ 。

下一步 观察向量  $X = [x_1, \cdots, x_M]^T$  由下式决定 注意  $M \geq N$ ：

$$X = AS + N \quad (7-188)$$

其中  $A$  是  $M \times N$  维矩阵。 $N = [n_1, \cdots, n_M]^T$  为加性噪声向量, 其各分量是均值为 0 的高斯随机变量且相互统计独立, 设分量  $n_i$  的方差是  $\nu_i^2$ , 则  $N$  的方差阵是一个  $M \times M$  维对角阵  $\Lambda$  即

$$\Lambda = E[NN^T] = \text{diag}[\nu_1^2, \cdots, \nu_M^2] \quad (7-189)$$

在已知  $S$  的条件下,  $X$  的条件概率密度记为  $\hat{p}(X | S, A, \Lambda)$ ,

$$\hat{p}(X | S, A, \Lambda) = (2\pi)^{-\frac{M}{2}} (\det \Lambda)^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (X - AS)^T \Lambda^{-1} (X - AS) \right\} \quad (7-190)$$

可以看到, 上列诸式描述的是一个随机向量产生链, 即

$q(N$  维不可见状态向量)  $\Rightarrow S(N$  维不可见源信号向量)  $\Rightarrow X(M$  维可见观察信号向量)  $X$  由此模型产生时 其 pdf 记为  $\hat{p}(X | \lambda)$ , 可称其为模型生成概率密度,  $\lambda$  是模型中所有参数的总和。此 pdf 为

$$\hat{p}(X | \lambda) = \int_S \hat{p}(S, X | \lambda) dS = \int_S \hat{p}(X | S, A, \Lambda) \hat{p}(S | \Omega, \xi) dS \quad (7-191)$$

其中  $\hat{p}(\mathbf{S}, \mathbf{X} | \boldsymbol{\lambda})$  是  $\mathbf{S}$  和  $\mathbf{X}$  的联合 pdf。引用式 (7-186) , 式 (7-191) 可写为

$$\hat{p}(\mathbf{X} | \boldsymbol{\lambda}) = \sum_q \hat{p}(\mathbf{q} | \boldsymbol{\Omega}) \int_{\mathbf{S}} d\mathbf{S} \cdot \hat{p}(\mathbf{X} | \mathbf{S}, \mathbf{A}, \boldsymbol{\Lambda}) \hat{p}(\mathbf{S} | \mathbf{q}, \boldsymbol{\xi}_q) \quad (7-192)$$

$\boldsymbol{\lambda}$  表示  $\{\mathbf{A}, \boldsymbol{\Lambda}$  全部  $\boldsymbol{\xi}_q, \boldsymbol{\Omega}\}$ 。

设观察向量  $\mathbf{X}$  的真实 pdf 是  $p(\mathbf{X})$  那么有噪 IFA 的算法框架包括估计  $\boldsymbol{\lambda}$  和求  $\mathbf{S}$  两部分。在第一部分, 先建立一个衡量  $\hat{p}(\mathbf{X} | \boldsymbol{\lambda})$  和  $p(\mathbf{X})$  之间差距的目标函数  $L(\boldsymbol{\lambda})$  然后用 EM 算法求出使  $L(\boldsymbol{\lambda})$  达最小的最优参数  $\boldsymbol{\lambda}$  从而建立一个最优  $\mathbf{X}$  生成模型。在第二部分则用此模型估计出未知的  $\mathbf{S}$ 。

### 7.11.2 有噪 IFA 的目标函数建立和求最优 $\boldsymbol{\lambda}$ 的 EM 算法

与 ICA 相同, IFA 也取两个 pdf 之间的 Kullback-Leibler 距离来衡量它们的差异, 因而可构成目标函数  $L(\boldsymbol{\lambda})$  如下 (参见式 (7-26)):

$$L(\boldsymbol{\lambda}) = \int_{\mathbf{X}} p(\mathbf{X}) \ln \frac{p(\mathbf{X})}{\hat{p}(\mathbf{X} | \boldsymbol{\lambda})} d\mathbf{X} = -H(\mathbf{X}) - \int_{\mathbf{X}} p(\mathbf{X}) \ln \hat{p}(\mathbf{X} | \boldsymbol{\lambda}) d\mathbf{X} \quad (7-193)$$

其中  $H(\mathbf{X}) = -\int_{\mathbf{X}} p(\mathbf{X}) \ln p(\mathbf{X}) d\mathbf{X}$  是  $\mathbf{X}$  的熵 因其与  $\boldsymbol{\lambda}$  无关 故可从 KL 距离中删除。在下面就采用  $H(\mathbf{X})$  被删后的 KL 距离作为目标函数, 即

$$L(\boldsymbol{\lambda}) = -E_{\mathbf{X}}[\ln \hat{p}(\mathbf{X} | \boldsymbol{\lambda})] = -\int_{\mathbf{X}} p(\mathbf{X}) \ln \hat{p}(\mathbf{X} | \boldsymbol{\lambda}) d\mathbf{X} \quad (7-194)$$

下面介绍的 EM 算法是一种异于按梯度计算最陡下降方向的迭代算法, 它具有更高的效率<sup>[49]</sup>。迭代按节拍  $k$  进行 设已求得  $\boldsymbol{\lambda}(k)$  则分两步求  $\boldsymbol{\lambda}(k+1)$ 。第一步 (E-步), 建立  $L(\boldsymbol{\lambda}(k+1))$  的上限函数  $\mathcal{F}(\boldsymbol{\lambda}(k+1), \boldsymbol{\lambda}(k))$ 。第二步 (M-步), 令  $\mathcal{F}(\cdot, \cdot)$  对  $\boldsymbol{\lambda}(k+1)$  的导数为 0 通过求得使  $\mathcal{F}(\cdot, \cdot)$  导数为 0 的  $\boldsymbol{\lambda}(k+1)$  从而使  $L(\boldsymbol{\lambda}(k+1))$  减至一尽量小的值。下面分别介绍这两步。

#### 1. E-步 expectation)

设节拍  $k$  时已获得参数  $\boldsymbol{\lambda}(k)$  基于此参数求  $k+1$  的参数  $\boldsymbol{\lambda}(k+1)$  使  $L(\boldsymbol{\lambda}(k+1))$  尽量小。为此 由式 (7-194) 和式 (7-192) 可得

$$\begin{aligned} L(\boldsymbol{\lambda}(k+1)) &= -E_{\mathbf{X}}[\ln \hat{p}(\mathbf{X} | \boldsymbol{\lambda}(k+1))] \\ &= -E_{\mathbf{X}}\left[\ln \sum_q \int_{\mathbf{S}} \hat{p}(\mathbf{q}, \mathbf{S}, \mathbf{X} | \boldsymbol{\lambda}(k+1)) d\mathbf{S}\right] \end{aligned} \quad (7-195)$$

其中

$$\hat{p}(\mathbf{q}, \mathbf{S}, \mathbf{X} | \boldsymbol{\lambda}(k+1)) = \hat{p}(\mathbf{q} | \boldsymbol{\Omega}(k+1)) \hat{p}(\mathbf{S} | \mathbf{q}, \boldsymbol{\xi}_q(k+1)) \hat{p}(\mathbf{X} | \mathbf{q}, \mathbf{S}, \mathbf{A}(k+1), \boldsymbol{\Lambda}(k+1))$$

下面推导中, 暂时略去对  $\mathbf{X}$  的集合取平均运算  $E_{\mathbf{X}}[\cdot]$  在求最终结果时再补入 利用对数函数的凸函数性质 可以证明下列不等式成立 注意 这是 EM 算法中的一个关键公式):

$$\begin{aligned} -\ln \sum_q \int_{\mathbf{S}} d\mathbf{S} \hat{p}(\mathbf{q}, \mathbf{S}, \mathbf{X} | \boldsymbol{\lambda}(k+1)) &\leq -\sum_q \int_{\mathbf{S}} \hat{p}(\mathbf{q}, \mathbf{S} | \mathbf{X}, \boldsymbol{\lambda}(k)) \ln \frac{\hat{p}(\mathbf{q}, \mathbf{S}, \mathbf{X} | \boldsymbol{\lambda}(k+1))}{\hat{p}(\mathbf{q}, \mathbf{S} | \mathbf{X}, \boldsymbol{\lambda}(k))} d\mathbf{S} \\ &= \mathcal{F}(\boldsymbol{\lambda}(k+1), \boldsymbol{\lambda}(k)) \end{aligned} \quad (7-196)$$

其中



$$\hat{p}(\mathbf{q}, \mathbf{S} | \mathbf{X}, \lambda(k)) = \hat{p}(\mathbf{q} | \mathbf{X}, \lambda(k)) \hat{p}(\mathbf{S} | \mathbf{q}, \mathbf{X}, \lambda(k))$$

此不等式称为 Jensen 不等式, 其证明可见文献 [44]。现在只要求得一最佳  $\lambda(k+1)$  使  $\mathcal{F}(\cdot, \cdot)$  达到极小, 则可使上列目标函数的上限极小。在此 E-步中先将  $\mathcal{F}$  计算出来。首先据式 (7-196),  $\mathcal{F}(\cdot, \cdot)$  可写成

$$\mathcal{F}(\lambda(k+1), \lambda(k)) = - \sum_{\mathbf{q}} \int_{\mathbf{S}} \hat{p}(\mathbf{q}, \mathbf{S} | \mathbf{X}, \lambda(k)) \ln \hat{p}(\mathbf{q}, \mathbf{S} | \lambda(k+1)) - H_{\mathbf{X}}(\mathbf{q}, \mathbf{S} | \lambda(k)) d\mathbf{S}$$

其中

$$H_{\mathbf{X}}(\mathbf{q}, \mathbf{S} | \lambda(k)) = - \sum_{\mathbf{q}} \int_{\mathbf{S}} \hat{p}(\mathbf{q}, \mathbf{S} | \mathbf{X}, \lambda(k)) \ln \hat{p}(\mathbf{q}, \mathbf{S} | \mathbf{X}, \lambda(k)) d\mathbf{S}$$

它与  $\lambda(k+1)$  无关 所以在求最佳  $\lambda(k+1)$  时将其略去。本式右侧第一项可分解为 3 项 用式 (7-195):  $\mathcal{F}_V(\cdot, \cdot)$ 、 $\mathcal{F}_B(\cdot, \cdot)$  和  $\mathcal{F}_T(\cdot, \cdot)$  其计算公式如下列。为使  $\mathcal{F}(\cdot, \cdot)$  达到极小, 应使这 3 个函数都达到极小。

$$\mathcal{F}_V(\lambda(k+1), \lambda(k)) = - \sum_{\mathbf{q}} \int_{\mathbf{S}} \hat{p}(\mathbf{q}, \mathbf{S} | \mathbf{X}, \lambda(k)) \ln \hat{p}(\mathbf{X} | \mathbf{q}, \mathbf{S}, \mathbf{A}(k+1), \mathbf{A}(k+1)) d\mathbf{S} \quad (7-197)$$

$$\mathcal{F}_B(\lambda(k+1), \lambda(k)) = - \sum_{\mathbf{q}} \int_{\mathbf{S}} \hat{p}(\mathbf{q}, \mathbf{S} | \mathbf{X}, \lambda(k)) \ln \hat{p}(\mathbf{S} | \mathbf{q}, \xi_{\mathbf{q}}(k+1)) d\mathbf{S} \quad (7-198)$$

$$\mathcal{F}_T(\lambda(k+1), \lambda(k)) = - \sum_{\mathbf{q}} \int_{\mathbf{S}} \hat{p}(\mathbf{q}, \mathbf{S} | \mathbf{X}, \lambda(k)) \ln \hat{p}(\mathbf{q} | \mathbf{S}, \mathbf{A}(k+1)) d\mathbf{S} \quad (7-199)$$

这 3 个函数的具体计算方法导出如下。

(1)  $\mathcal{F}_V(\cdot, \cdot)$  的计算

根据式 (7-190) 可以导出

$$\begin{aligned} -\ln \hat{p}(\mathbf{X} | \mathbf{S}, \mathbf{A}(k+1), \mathbf{A}(k+1)) &= \frac{M}{2} \ln 2\pi + \frac{1}{2} \ln |\det \mathbf{A}(k+1)| + \frac{1}{2} [\mathbf{X}^T \mathbf{A}^{-1} \cdot \\ &\quad (\mathbf{k}+1) \mathbf{X} - 2 \mathbf{S}^T \mathbf{A}^T (\mathbf{k}+1) \mathbf{A}^{-1} (\mathbf{k}+1) \mathbf{X} + \\ &\quad \mathbf{S}^T \mathbf{A}^T (\mathbf{k}+1) \mathbf{A}^{-1} (\mathbf{k}+1) \mathbf{A} (\mathbf{k}+1) \mathbf{S}] \end{aligned}$$

根据式 (7-46) 关于矩阵迹的定义, 此式右侧第 3 项可表示为

$$\frac{1}{2} \text{tr} \{ \mathbf{A}^{-1} (\mathbf{k}+1) [\mathbf{X} \mathbf{X}^T - 2 \mathbf{X} \mathbf{S}^T \mathbf{A}^T (\mathbf{k}+1) + \mathbf{A} (\mathbf{k}+1) \mathbf{S} \mathbf{S}^T \mathbf{A}^T (\mathbf{k}+1)] \}$$

此式右侧第 1 项为常数, 与求极小无关, 故可略去。将此式代入式 (7-197) 并且注意到下式成立:

$$\sum_{\mathbf{q}} \hat{p}(\mathbf{q}, \mathbf{S} | \mathbf{X}, \lambda(k)) = \hat{p}(\mathbf{S} | \mathbf{X}, \lambda(k)) \quad (7-200)$$

则可得

$$\begin{aligned} \mathcal{F}_V(\lambda(k+1), \lambda(k)) &= \frac{1}{2} \ln |\det \mathbf{A}(k+1)| + \frac{1}{2} \text{tr} \{ \mathbf{A}^{-1} (\mathbf{k}+1) [\mathbf{X} \mathbf{X}^T - \\ &\quad 2 \mathbf{X} \langle \mathbf{S}^T | \mathbf{X}, \mathbf{k} \rangle \mathbf{A}^T (\mathbf{k}+1) + \mathbf{A} (\mathbf{k}+1) \langle \mathbf{S} \mathbf{S}^T | \mathbf{X}, \mathbf{k} \rangle \mathbf{A}^T (\mathbf{k}+1)] \} \end{aligned} \quad (7-201)$$

其中  $\langle \mathbf{S}^T | \mathbf{X}, \mathbf{k} \rangle$  和  $\langle \mathbf{S} \mathbf{S}^T | \mathbf{X}, \mathbf{k} \rangle$  用下式计算 设  $\boldsymbol{\psi} = \mathbf{S}^T$  或  $\mathbf{S} \mathbf{S}^T$  则

$$\langle \psi | \mathbf{X}, k \rangle = \int_S \hat{p}(\mathbf{S} | \mathbf{X}, \lambda(k)) \psi d\mathbf{S} \quad (7-202)$$

为计算  $\langle \psi \rangle$  需求得  $\hat{p}(\mathbf{S} | \mathbf{X}, \lambda(k))$ 。按照式 (7-200),

$$\hat{p}(\mathbf{S} | \mathbf{X}, \lambda(k)) = \sum_q \hat{p}(\mathbf{q}, \mathbf{S} | \mathbf{X}, \lambda(k)) = \sum_q \hat{p}(\mathbf{q} | \mathbf{X}, \lambda(k)) \hat{p}(\mathbf{S} | \mathbf{q}, \mathbf{X}, \lambda(k)) \quad (7-203)$$

此式右侧和式中的两个条件概率可计算如下 (用贝叶斯公式) :

$$\hat{p}(\mathbf{q} | \mathbf{X}, \lambda(k)) = \frac{\hat{p}(\mathbf{q} | \lambda(k)) \hat{p}(\mathbf{X} | \mathbf{q}, \lambda(k))}{\hat{p}(\mathbf{X} | \lambda(k))} \quad (7-204)$$

其中  $\hat{p}(\mathbf{q} | \lambda(k)) = \prod_{i=1}^N \omega_{iq_i}(k)$  是在节拍  $k$  求得的。其次,

$$\hat{p}(\mathbf{X} | \mathbf{q}, \lambda(k)) = \int_S \hat{p}(\mathbf{X} | \mathbf{S}, \lambda(k)) \hat{p}(\mathbf{S} | \mathbf{q}, \lambda(k)) d\mathbf{S}$$

将式 (7-187) 和 (7-190) 代入上式并作简单推导, 即得

$$\hat{p}(\mathbf{X} | \mathbf{q}, \lambda(k)) = (2\pi)^{-\frac{M}{2}} \left| \det \boldsymbol{\Sigma}_{\mathbf{X}|\mathbf{q}} \right|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (\mathbf{X} - \mathbf{A}\boldsymbol{\mu}_q)^T \boldsymbol{\Sigma}_{\mathbf{X}|\mathbf{q}}^{-1} (\mathbf{X} - \mathbf{A}\boldsymbol{\mu}_q) \right\}$$

$\boldsymbol{\Sigma}_{\mathbf{X}|\mathbf{q}} = \mathbf{A}\mathbf{V}_q\mathbf{A}^T + \boldsymbol{\Lambda}$ ,  $(\mathbf{A}, \boldsymbol{\Lambda}, \boldsymbol{\mu}_q, \mathbf{V}_q)$  等参数是节拍  $k$  所求得参数, 为简化而未写成  $\mathbf{A}(k)$ ,  $\boldsymbol{\Lambda}(k), \dots$  (7-205)

$$\hat{p}(\mathbf{X} | \lambda(k)) = \sum_q \hat{p}(\mathbf{q}, \mathbf{X} | \lambda(k)) = \sum_q \hat{p}(\mathbf{q} | \lambda(k)) \hat{p}(\mathbf{X} | \mathbf{q}, \lambda(k)) \quad (7-206)$$

另一个概率也可用贝叶斯公式计算如下:

$$\hat{p}(\mathbf{S} | \mathbf{q}, \mathbf{X}, \lambda(k)) = \frac{\hat{p}(\mathbf{S} | \mathbf{q}, \lambda(k)) \hat{p}(\mathbf{X} | \mathbf{q}, \mathbf{S}, \lambda(k))}{\hat{p}(\mathbf{X} | \mathbf{q}, \lambda(k))}$$

其中  $\hat{p}(\mathbf{S} | \mathbf{q}, \lambda(k))$  可用式 (7-187) 代入,  $\hat{p}(\mathbf{X} | \mathbf{q}, \lambda(k))$  可用式 (7-205) 代入; 此外, 在  $\mathbf{S}$  给定的条件下  $\mathbf{q}$  不再起作用, 这时  $\hat{p}(\mathbf{X} | \mathbf{q}, \mathbf{S}, \lambda(k)) = \hat{p}(\mathbf{X} | \mathbf{S}, \lambda(k))$  后者可用式 (7-190) 代入计算。经推导可得<sup>[41]</sup>

$$\begin{aligned} \hat{p}(\mathbf{S} | \mathbf{q}, \mathbf{X}, \lambda(k)) &= (2\pi)^{-\frac{N}{2}} (\boldsymbol{\Sigma}_q)^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (\mathbf{S} - \boldsymbol{\zeta}_q(\mathbf{X}))^T (\boldsymbol{\Sigma}_q)^{-1} (\mathbf{S} - \boldsymbol{\zeta}_q(\mathbf{X})) \right\} \\ \boldsymbol{\Sigma}_q &= [\mathbf{A}^T \boldsymbol{\Lambda}^{-1} \mathbf{A} + \mathbf{V}_q^{-1}]^{-1}, \quad \boldsymbol{\zeta}_q(\mathbf{X}) = \boldsymbol{\Sigma}_q [\mathbf{A}^T \boldsymbol{\Lambda}^{-1} \mathbf{X} + \mathbf{V}_q^{-1} \boldsymbol{\mu}_q] \end{aligned} \quad (7-207)$$

注意,  $\mathbf{A}, \boldsymbol{\Lambda}, \mathbf{V}_q, \boldsymbol{\mu}_q$  都是节拍  $k$  的参数, 为简化而未标明。 $\boldsymbol{\Sigma}_q$  为  $N \times N$  维,  $\boldsymbol{\zeta}_q(\mathbf{X})$  为  $N \times 1$  维, 将式 (7-207) 代入式 (7-203) 再将式 (7-203) 代入式 (7-202) 即可求得

$$\left. \begin{aligned} \langle \mathbf{S}^T | \mathbf{X}, k \rangle &= \sum_q \hat{p}(\mathbf{q} | \mathbf{X}, \lambda(k)) \langle \mathbf{S}^T | \mathbf{q}, \mathbf{X}, k \rangle \\ \langle \mathbf{S}\mathbf{S}^T | \mathbf{X}, k \rangle &= \sum_q \hat{p}(\mathbf{q} | \mathbf{X}, \lambda(k)) \langle \mathbf{S}\mathbf{S}^T | \mathbf{q}, \mathbf{X}, k \rangle \\ \langle \mathbf{S}^T | \mathbf{q}, \mathbf{X}, k \rangle &= \boldsymbol{\zeta}_q^T(\mathbf{X}) \\ \langle \mathbf{S}\mathbf{S}^T | \mathbf{q}, \mathbf{X}, k \rangle &= \boldsymbol{\Sigma}_q + \boldsymbol{\zeta}_q(\mathbf{X}) \boldsymbol{\zeta}_q^T(\mathbf{X}) \end{aligned} \right\} \quad (7-208)$$

最后, 恢复式 (7-195) 中的  $E_x[\cdot]$  则式 (7-201) 中的  $\mathbf{X}\mathbf{X}^T, \mathbf{X}\langle \mathbf{S}^T | \mathbf{X}, k \rangle$  和  $\langle \mathbf{S}\mathbf{S}^T | \mathbf{X}, k \rangle$  应该用  $\langle \mathbf{X}\mathbf{X}^T \rangle = E_x[\mathbf{X}\mathbf{X}^T]$ ,  $\langle \mathbf{X}\langle \mathbf{S}^T | \mathbf{X}, k \rangle \rangle = E_x[\mathbf{X}\langle \mathbf{S}^T | \mathbf{X}, k \rangle]$  和  $\langle \mathbf{S}\mathbf{S}^T | \mathbf{X}, k \rangle = E_x[\langle \mathbf{S}\mathbf{S}^T |$

$\mathbf{X}, k\rangle]$  来取代。在实际运算时 对  $\mathbf{X}$  的集合平均  $E_{\mathbf{X}}[\cdot]$  不可能实现, 应用有限样本的批平均代之 (见式 7-31))。

(2)  $\mathcal{F}_B(\cdot, \cdot)$  的计算

根据式 (7-198) ,式(7-200) ,式( 7-201) 得

$$\mathcal{F}_B(\boldsymbol{\lambda}(k+1), \boldsymbol{\lambda}(k)) = - \sum_{\mathbf{q}} \hat{p}(\mathbf{q} | \mathbf{X}, \boldsymbol{\lambda}(k)) \int_S \hat{p}(\mathbf{S} | \mathbf{q}, \mathbf{X}, \boldsymbol{\lambda}(k)) \ln \hat{p}(\mathbf{S} | \mathbf{q}, \boldsymbol{\xi}_{\mathbf{q}}(k+1)) d\mathbf{S} \quad (7-209)$$

由式 (7-187) 得到

$$\begin{aligned} \ln \hat{p}(\mathbf{S} | \mathbf{q}, \boldsymbol{\xi}_{\mathbf{q}}(k+1)) &= \sum_{i=1}^N \ln \hat{p}_i(S_i | q_i, \boldsymbol{\xi}_{\mathbf{q}}(k+1)) \\ n \hat{p}_i(S_i | q_i, \boldsymbol{\xi}_{\mathbf{q}}(k+1)) &= -\frac{1}{2} \ln 2\pi - \ln \sigma_{iq_i}(k+1) - \frac{1}{2\sigma_{iq_i}^2(k+1)} \cdot \\ &\quad [S_i^2 - 2S_i \mu_{iq_i}(k+1) + \mu_{iq_i}^2(k+1)] \end{aligned}$$

如果略去与  $\boldsymbol{\lambda}(k+1)$  无关的常数项,  $-\frac{1}{2} \ln 2\pi$  且注意到

$$\sum_{\mathbf{q}} = \sum_{q_1=1}^{q_1} \cdots \sum_{q_N=1}^{q_N}$$

则式 (7-209) 可以表示为

$$\begin{aligned} \mathcal{F}_B(\boldsymbol{\lambda}(k+1), \boldsymbol{\lambda}(k)) &= \sum_{i=1}^N \sum_{q_i=1}^{q_i} \hat{p}_i(q_i | \mathbf{X}, \boldsymbol{\lambda}(k)) \left\{ \ln \sigma_{iq_i}(k+1) + \frac{1}{2\sigma_{iq_i}^2(k+1)} \cdot \right. \\ &\quad \left. [\langle S_i^2 | q_i, \mathbf{X} \rangle - 2\langle S_i | q_i, \mathbf{X} \rangle \mu_{iq_i}(k+1) + \mu_{iq_i}^2(k+1)] \right\} \quad (7-210) \end{aligned}$$

其中

$$\hat{p}_i(q_i | \mathbf{X}, \boldsymbol{\lambda}(k)) = \sum_{q_1=1}^{q_1} \cdots \sum_{\substack{q_j=1 \\ q_j \neq q_i}}^{q_j} \cdots \sum_{q_N=1}^{q_N} \hat{p}(\mathbf{q} | \mathbf{X}, \boldsymbol{\lambda}(k)) \quad (7-211)$$

$\langle S_i^2 | q_i, \mathbf{X} \rangle$  是下列矩阵的第  $i$  个对角元素,

$$\langle \mathbf{S} \mathbf{S}^T | q_i, \mathbf{X} \rangle = \frac{\sum_{q_1=1}^{q_1} \cdots \sum_{\substack{q_j=1 \\ q_j \neq q_i}}^{q_j} \cdots \sum_{q_N=1}^{q_N} \hat{p}(\mathbf{q} | \mathbf{X}, \boldsymbol{\lambda}(k)) \langle \mathbf{S} \mathbf{S}^T | \mathbf{q}, \mathbf{X} \rangle}{\hat{p}_i(q_i | \mathbf{X}, \boldsymbol{\lambda}(k))} \quad (7-212)$$

$\langle S_i | q_i, \mathbf{X} \rangle$  是下列向量的第  $i$  个元素,

$$\langle \mathbf{S}^T | q_i, \mathbf{X} \rangle = \frac{\sum_{q_1=1}^{q_1} \cdots \sum_{\substack{q_j=1 \\ q_j \neq q_i}}^{q_j} \cdots \sum_{q_N=1}^{q_N} \hat{p}(\mathbf{q} | \mathbf{X}, \boldsymbol{\lambda}(k)) \langle \mathbf{S}^T | \mathbf{q}, \mathbf{X} \rangle}{\hat{p}_i(q_i | \mathbf{X}, \boldsymbol{\lambda}(k))} \quad (7-213)$$

最后 恢复  $E_{\mathbf{X}}[\cdot]$  如定义:

$$\begin{aligned} E_{\mathbf{X}}[\hat{p}_i(q_i | \mathbf{X}, \boldsymbol{\lambda}(k))] &= \langle \hat{p}_i(q_i | \mathbf{X}, k) \rangle \\ E_{\mathbf{X}}[\hat{p}_i(q_i | \mathbf{X}, \boldsymbol{\lambda}(k)) \langle S_i^2 | q_i, \mathbf{X} \rangle] &= \langle S_i^2 | q_i, \mathbf{X}, k \rangle \\ E_{\mathbf{X}}[\hat{p}_i(q_i | \mathbf{X}, \boldsymbol{\lambda}(k)) \langle S_i | q_i, \mathbf{X} \rangle] &= \langle S_i | q_i, \mathbf{X}, k \rangle \end{aligned} \quad (7-214)$$

则式 (7-210) 可表示为

$$\begin{aligned}\mathcal{F}_B(\lambda(k+1), \lambda(k)) = & \sum_{i=1}^N \sum_{q_i=1}^{Q_i} \langle \hat{p}_i(q_i | \mathbf{X}, k) \rangle \ln \sigma_{i q_i}(k+1) + \frac{1}{2\sigma_{i q_i}^2(k+1)} \cdot \\ & \{ \langle S_i^2 | q_i, \mathbf{X}, k \rangle - 2\langle S_i | q_i, \mathbf{X}, k \rangle \mu_{i q_i}(k+1) + \\ & \langle \hat{p}_i(q_i | \mathbf{X}, k) \rangle \mu_{i q_i}^2(k+1) \}\end{aligned}\quad (7-215)$$

在实际运算中  $E_x[\cdot]$  用有限样本批平均取代。

(3)  $\mathcal{F}_T(\cdot, \cdot)$  的计算

根据式 (7-184)

$$\ln \hat{p}(\mathbf{q} | \boldsymbol{\Omega}(k+1)) = \sum_{i=1}^N \ln \omega_{i q_i}(k+1)$$

将其代入式 (7-199) 并按前述推导 即可得

$$\mathcal{F}_T(\lambda(k+1), \lambda(k)) = \sum_{i=1}^N \sum_{q_i=1}^{Q_i} \hat{p}_i(q_i | \mathbf{X}, \lambda(k)) \ln \omega_{i q_i}(k+1) \quad (7-216)$$

当恢复  $E_x[\cdot]$  时  $\hat{p}_i(q_i | \mathbf{X}, \lambda(k))$  用  $\langle \hat{p}_i(q_i | \mathbf{X}, k) \rangle$  取代 (见式 7-214))。

## 2. M-步 (minimization)

基于 E-步求得的  $\mathcal{F}_V, \mathcal{F}_B, \mathcal{F}_T$  求它们相对于  $\lambda(k+1)$  中诸参数的导数, 这些导数的 0 值点即决定了  $k+1$  步时的最优参数。

首先  $\mathcal{F}_A(\cdot, \cdot)$  相对于  $\mathbf{A}(k+1)$  和  $\boldsymbol{\Lambda}(k+1)$  的导数由式 (7-201) 求得如下:

$$\frac{\partial \mathcal{F}_V}{\partial \mathbf{A}(k+1)} = -\mathbf{A}^{-1}(k+1) \langle \mathbf{X} \langle \mathbf{S}^T | \mathbf{X}, k \rangle \rangle + \mathbf{A}^{-1}(k+1) \mathbf{A}(k+1) \langle \mathbf{S} \mathbf{S}^T | \mathbf{X}, k \rangle \quad (2-217)$$

$$\begin{aligned}\frac{\partial \mathcal{F}_V}{\partial \boldsymbol{\Lambda}(k+1)} = & \frac{1}{2} \mathbf{A}^{-1}(k+1) - \frac{1}{2} \mathbf{A}^{-1}(k+1) \{ \langle \mathbf{X} \mathbf{X}^T \rangle - 2\langle \mathbf{X} \langle \mathbf{S}^T | \mathbf{X}, k \rangle \rangle \mathbf{A}^T(k+1) + \\ & \mathbf{A}(k+1) \langle \mathbf{S} \mathbf{S}^T | \mathbf{X}, k \rangle \rangle \mathbf{A}^T(k+1) \} \mathbf{A}^{-1}(k+1)\end{aligned}\quad (7-218)$$

令式 (7-217) 和式 (7-218) 右侧为 0 即求得  $k+1$  节拍的最佳  $\mathbf{A}$  和  $\boldsymbol{\Lambda}$  为

$$\mathbf{A}(k+1) = \langle \mathbf{X} \langle \mathbf{S}^T | \mathbf{X}, k \rangle \rangle \langle \mathbf{S} \mathbf{S}^T | \mathbf{X}, k \rangle^{-1} \quad (7-219)$$

$$\boldsymbol{\Lambda}(k+1) = \langle \mathbf{X} \mathbf{X}^T \rangle - \langle \mathbf{X} \langle \mathbf{S}^T | \mathbf{X}, k \rangle \rangle \mathbf{A}^T(k+1) \quad (7-220)$$

其次 令式 (7-215) 中的  $\sigma_{i q_i}^2 = \nu_{i q_i}$  则  $\mathcal{F}_B(\cdot, \cdot)$  对各  $\nu_{i q_i}$  和  $\mu_{i q_i}$  的导数如下列:

$$\frac{\partial \mathcal{F}_B}{\partial \mu_{i q_i}(k+1)} = -\frac{1}{\nu_{i q_i}(k+1)} \{ \langle S_i | q_i, \mathbf{X}, k \rangle - \langle \hat{p}_i(q_i | \mathbf{X}, k) \rangle \mu_{i q_i}(k+1) \} \quad (7-221)$$

$$\begin{aligned}\frac{\partial \mathcal{F}_B}{\partial \nu_{i q_i}(k+1)} = & -\frac{1}{2\nu_{i q_i}^2(k+1)} \{ \langle S_i^2 | q_i, \mathbf{X}, k \rangle - 2\langle S_i | q_i, \mathbf{X}, k \rangle \mu_{i q_i}(k+1) + \\ & \langle \hat{p}_i(q_i | \mathbf{X}, k) \rangle (\mu_{i q_i}^2(k+1) - \nu_{i q_i}(k+1)) \}\end{aligned}\quad (7-222)$$

令式 (7-221) 和式 (7-222) 右侧等于 0 可求得节拍  $k+1$  的各最佳  $\mu_{i q_i}(k+1)$  和  $\nu_{i q_i}(k+1) = \sigma_{i q_i}^2(k+1)$  为

$$\mu_{i q_i}(k+1) = \frac{\langle S_i | q_i, \mathbf{X}, k \rangle}{\langle \hat{p}_i(q_i | \mathbf{X}, k) \rangle}, \quad i = 1 \sim N, \quad q_i = 1 \sim Q_i \quad (7-223)$$

$$\nu_{i q_i}(k+1) = \sigma_{i q_i}^2(k+1) = \frac{\langle S_i^2 | q_i, \mathbf{X}, k \rangle}{\langle \hat{p}_i(q_i | \mathbf{X}, k) \rangle} - \mu_{i q_i}^2(k+1),$$

$$i = 1 \sim N, \quad q_i = 1 \sim Q_i \quad (7-224)$$

最后, 为求式 (7-216) 给出的  $\mathcal{F}_T(\cdot, \cdot)$  达到最小的诸  $\omega_{iq_i}$  且满足  $\sum_{q_i=1}^{Q_i} \omega_{iq_i} = 1$  作如下变换:

$$\omega_{iq_i} = \frac{\exp \bar{\omega}_{iq_i}}{\sum_{q_i=1}^{Q_i} \exp \bar{\omega}_{iq'_i}}, \quad q_i = 1 \sim Q_i \quad (7-225)$$

将式 (7-225) 代入式 (7-216) 并令  $\mathcal{F}_T(\cdot, \cdot)$  对诸  $\bar{\omega}_{iq_i}$  的导数为 0 即可求得  $k+1$  节拍的各最佳  $\omega_{iq_i}(k+1)$  为

$$\omega_{iq_i}(k+1) = \langle \hat{p}_i(q_i | \mathbf{X}, k) \rangle, \quad i = 1 \sim N, \quad q_i = 1 \sim Q_i \quad (7-226)$$

最后 求得了各  $\nu_{iq_i}(k+1)$ ,  $\mu_{iq_i}(k+1)$  和  $\omega_{iq_i}(k+1)$  以后, 可以按照式 (7-182) 求得各  $S_i$  按模型产生时的均方误差  $\hat{\sigma}_i^2(k+1)$  是

$$\begin{aligned} \hat{\sigma}_i^2(k+1) &= \int_{S_i} \hat{p}_i(S_i | \boldsymbol{\Omega}_i(k+1), \boldsymbol{\xi}_i(k+1)) S_i^2 dS_i - \left[ \int_{S_i} \hat{p}_i(S_i | \boldsymbol{\Omega}_i(k+1), \boldsymbol{\xi}_i(k+1)) S_i dS_i \right]^2 \\ &= \sum_{q_i=1}^{Q_i} \omega_{iq_i}(k+1) (\nu_{iq_i}(k+1) + \mu_{iq_i}^2(k+1)) - \left[ \sum_{q_i=1}^{Q_i} \omega_{iq_i}(k+1) \mu_{iq_i}(k+1) \right]^2 \end{aligned} \quad (7-227)$$

各  $\hat{\sigma}^2(k+1)$  的不一致会使算法的效率降低<sup>[41]</sup> 为此可以采用归一化方法 使各  $S_i$  的模型产生均方误差都等于 1 而不改变模型产生观察向量  $\mathbf{X}$  的 pdf。采用 7.2.2 小节所述的等价尺度变换就能做到这一点。这时将 EM 算法得到的各个  $\mu_{jq_j}(k+1)$ ,  $\nu_{jq_j}(k+1)$  和  $a_{ij}(k+1)$  ( $\mathbf{A}(k+1)$  的第  $i$  行  $j$  列元素) 按下列规则进行置换:

$$\left. \begin{aligned} \mu_{jq_j}(k+1) &\rightarrow \mu_{jq_j}(k+1)/\hat{\sigma}_j(k+1), \quad q_j = 1 \sim Q_j, j = 1 \sim N \\ \nu_{jq_j}(k+1) &\rightarrow \nu_{jq_j}(k+1)/\hat{\sigma}_j(k+1), \quad q_j = 1 \sim Q_j, j = 1 \sim N \\ a_{ij}(k+1) &\rightarrow \hat{\sigma}_j(k+1)a_{ij}(k+1), \quad j = 1 \sim N, i = 1 \sim M \end{aligned} \right\} \quad (7-228)$$

### 7.11.3 有噪 IFA 源信号的恢复与模拟实验结果

在 7.11.2 小节中给出了用 EM 算法估计最佳生成模型参数的程序。下一步即需用这些参数来由观察向量  $\mathbf{X}$  对源信号向量  $\mathbf{S}$  作出最佳估计。先讨论一种特殊情况 —— 无噪声 ( $\mathbf{N} = \mathbf{0}$ ) 由式 (7-188) 可得  $\mathbf{X} = \mathbf{A}\mathbf{S}$ 。若  $M \neq N$  不能简单地用逆矩阵从  $\mathbf{X}$  求得  $\mathbf{S}$  为此可通过下面的运算实现由  $\mathbf{X}$  求  $\mathbf{S}$ :

$$\mathbf{A}^T \mathbf{X} = \mathbf{A}^T \mathbf{A} \mathbf{S}, \quad \mathbf{S} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{X} \quad (7-229)$$

由于  $\mathbf{A}^T \mathbf{A}$  是方阵, 若其非奇可作求逆运算。式 (7-229) 完成的由  $\mathbf{X}$  求  $\mathbf{S}$  过程称为拟逆 (pseudo-inverse) 关系。当  $M = N$  时, 此关系即退化为求逆过程:  $\mathbf{S} = \mathbf{A}^{-1} \mathbf{X}$ 。但是 在此特殊情况下求得的  $\mathbf{S}$  是不可信的。在 7.11.4 小节将证明 当  $\mathbf{N} = \mathbf{0}$  时有噪 IFA 算法将退化为 PCA 算法。在 7.2.4 小节的 2 中已指出 PCA 只去除二阶相关性而不去除高阶相关性, 因此不可能实现源信号恢复。7.11.4 小节还将说明, 即使  $\mathbf{N} \neq \mathbf{0}$ , 若其模值很小则有噪 IFA 算法脱离 PCA 算法的速度很慢。换言之 在  $\mathbf{N} = \mathbf{0}$  或  $\mathbf{N}$  的模值较小的情况 不能用式

(7-229) 给出正确的源信号向量  $\mathbf{S}$  (在 7.11.4 小节将给出针对此情况的无噪 IFA 算法)。在  $\mathbf{N}$  的模值较大的情况下, 由式 7-188) 可得  $\mathbf{X} = \mathbf{A}\mathbf{S} + \mathbf{N}$ , 这时可以按最小均方 (LMS) 准则或最大后验 (MAP) 准则由  $\mathbf{X}$  估计出  $\mathbf{S}$ 。

### 1. LMS 准则估计

设已知  $\mathbf{X}$  条件下的真实源信号为  $\mathbf{S}$  其 pdf,  $\hat{p}(\mathbf{S} | \mathbf{X}, \boldsymbol{\lambda})$  由式 (7-203) 给出 (这里用  $\boldsymbol{\lambda}$  代替  $\boldsymbol{\lambda}(k)$  表示已经求得的最佳参数) 设  $\mathbf{S}$  的估值为  $\hat{\mathbf{S}}$  则 LMS 准则要求  $\hat{\mathbf{S}}$  与  $\mathbf{S}$  之间的均方误差  $E[\|\mathbf{S} - \hat{\mathbf{S}}\|^2]$  最小。它可用下式计算:

$$E[\|\hat{\mathbf{S}} - \mathbf{S}\|^2] = \int_{\mathbf{S}} \|\hat{\mathbf{S}} - \mathbf{S}\|^2 \hat{p}(\mathbf{S} | \mathbf{X}, \boldsymbol{\lambda}) d\mathbf{S}$$

易于证明 当  $\mathbf{S} = E[\mathbf{S}]$  时 即可使  $E[\|\mathbf{S} - \hat{\mathbf{S}}\|^2]$  最小。此估值表示为  $\mathbf{S}_{\text{LMS}}$  且可用下式计算 (引式 7-203)):

$$\hat{\mathbf{S}}_{\text{LMS}} = E[\mathbf{S}] = \int_{\mathbf{S}} \hat{p}(\mathbf{S} | \mathbf{X}, \boldsymbol{\lambda}) \mathbf{S} d\mathbf{S} = \sum_{\mathbf{q}} \hat{p}(\mathbf{q} | \mathbf{X}, \boldsymbol{\lambda}) \int_{\mathbf{S}} \hat{p}(\mathbf{S} | \mathbf{q}, \mathbf{X}, \boldsymbol{\lambda}) \mathbf{S} d\mathbf{S}$$

再引用式 7-207) 即可求得

$$\hat{\mathbf{S}}_{\text{LMS}} = \sum_{\mathbf{q}} \hat{p}(\mathbf{q} | \mathbf{X}, \boldsymbol{\lambda}) \sum_{\mathbf{q}} [\mathbf{A}^T \boldsymbol{\Lambda}^{-1} \mathbf{X} + \mathbf{V}_{\mathbf{q}}^{-1} \boldsymbol{\mu}_{\mathbf{q}}] \quad (7-230)$$

其中  $\hat{p}(\mathbf{q} | \mathbf{X}, \boldsymbol{\lambda})$  可引式 7-204) 计算。

### 2. MAP 准则估计

按此准则得到的源信号估值记为  $\mathbf{S}_{\text{MAP}}$  它应使已知  $\mathbf{X}$  条件下的后验概率对数值达到最大 即

$$\mathbf{S}_{\text{MAP}} = \underset{\mathbf{S}}{\operatorname{argmax}} [\ln \hat{p}(\mathbf{S} | \mathbf{X}, \boldsymbol{\lambda})]$$

注意到  $\hat{p}(\mathbf{S} | \mathbf{X}, \boldsymbol{\lambda}) = \hat{p}(\mathbf{S} | \boldsymbol{\lambda}) \hat{p}(\mathbf{X} | \mathbf{S}, \boldsymbol{\lambda}) / \hat{p}(\mathbf{X} | \boldsymbol{\lambda})$  且  $\hat{p}(\mathbf{X} | \boldsymbol{\lambda})$  与  $\mathbf{S}$  无关 则

$$\mathbf{S}_{\text{MAP}} = \underset{\mathbf{S}}{\operatorname{argmax}} [\ln \hat{p}(\mathbf{X} | \mathbf{S}, \boldsymbol{\lambda}) + \ln \hat{p}(\mathbf{S} | \boldsymbol{\lambda})] \quad (7-231)$$

引用式 7-190) 得到 (注意,  $\boldsymbol{\lambda}$  含  $\mathbf{A}, \boldsymbol{\Lambda}, C$  是一个常数)

$$\ln \hat{p}(\mathbf{X} | \mathbf{S}, \boldsymbol{\lambda}) = C - \frac{1}{2} (\mathbf{X} - \mathbf{A}\mathbf{S})^T \boldsymbol{\Lambda}^{-1} (\mathbf{X} - \mathbf{A}\mathbf{S})$$

引用式 7-183) 得到 (注意,  $\boldsymbol{\lambda}$  含  $\boldsymbol{\Omega}, \boldsymbol{\xi}$ )

$$\ln \hat{p}(\mathbf{S} | \boldsymbol{\lambda}) = \sum_{i=1}^N \ln \hat{p}_i(S_i | \boldsymbol{\Omega}_i, \boldsymbol{\xi}_i)$$

其中  $\hat{p}_i(S_i | \boldsymbol{\Omega}_i, \boldsymbol{\xi}_i)$  由式 7-182) 计算。

由于式 (7-231) 右侧是  $\mathbf{S}$  的非线性函数, 通过解方程的方法直接求  $\hat{\mathbf{S}}_{\text{MAP}}$  是困难的。只能用迭代算法按节拍  $k$  对  $\mathbf{S}(k)$  进行计算。设初值为  $\mathbf{S}(0)$  则对于  $k = 0, 1, 2, \dots$  按下式由  $\mathbf{S}(k)$  求  $\mathbf{S}(k+1)$ :

$$\left. \begin{aligned} \hat{\mathbf{S}}(k+1) &= \hat{\mathbf{S}}(k) + \Delta \hat{\mathbf{S}}(k) \\ \Delta \hat{\mathbf{S}}(k) &= \alpha(k) [\nabla_{\mathbf{S}} \ln \hat{p}(\mathbf{X} | \mathbf{S}, \boldsymbol{\lambda}) + \nabla_{\mathbf{S}} \ln \hat{p}(\mathbf{S} | \boldsymbol{\lambda})] \Big|_{\mathbf{S} = \hat{\mathbf{S}}(k)} \end{aligned} \right\} \quad (7-232)$$

其中  $\alpha(k) > 0$  是一个随  $k$  的增加而逐渐缩小的步幅函数。式中的两个梯度为

$$\nabla_{\mathbf{S}} \ln \hat{p}(\mathbf{X} | \mathbf{S}, \boldsymbol{\lambda}) = \mathbf{A}^T \mathbf{A}^{-1} (\mathbf{X} - \mathbf{A}\mathbf{S}) \quad (7-233)$$

$$\left. \begin{aligned} \nabla_{\mathbf{S}} \ln \hat{p}(\mathbf{S} | \boldsymbol{\lambda}) &= \left[ \frac{\partial \ln \hat{p}_1(S_1 | \boldsymbol{\Omega}_1, \boldsymbol{\xi}_1)}{\partial S_1}, \dots, \frac{\partial \ln \hat{p}_N(S_N | \boldsymbol{\Omega}_N, \boldsymbol{\xi}_N)}{\partial S_N} \right]^T \\ \frac{\partial \ln \hat{p}_i(S_i | \boldsymbol{\Omega}_i, \boldsymbol{\xi}_i)}{\partial S_i} &= \sum_{q_i=1}^{Q_i} \hat{p}_i(q_i | S_i) \frac{S_i - \mu_{iq_i}}{\sigma_{iq_i}^2} \\ \hat{p}_i(q_i | S_i) &= \frac{\hat{p}_i(q_i) \hat{p}_i(S_i | q_i, \boldsymbol{\xi}_i)}{\hat{p}_i(S_i | \boldsymbol{\Omega}_i, \boldsymbol{\xi}_i)}, \quad i = 1 \sim N \end{aligned} \right\} \quad (7-234)$$

其中

$$\hat{p}_i(S_i | q_i, \boldsymbol{\xi}_i) = \frac{1}{\sqrt{2\pi\sigma_{iq_i}^2}} \exp \left\{ -\frac{1}{2} \left( \frac{S_i - \mu_{iq_i}}{\sigma_{iq_i}} \right)^2 \right\}$$

如迭代计算结果使  $\mathbf{S}$  收敛于  $\ln \hat{p}(\mathbf{S} | \mathbf{X}, \boldsymbol{\lambda})$  的一个极大点 (如有多个极大点时 则应取其中的较大者) 即可以之作为  $\mathbf{S}_{\text{MAP}}$ 。尽管迭代的初值  $\mathbf{S}(0)$  可选为式 7-229 给出的拟逆解 效果较好。但是, 最好从其他几个随机初值出发求得多个解, 以便从中选择最佳者。

文献 [41] 针对音乐信号的分离进行了实验, 条件是  $N = 3, M = 3$  或 8 在实验中取  $Q_i = 3, \forall i$ 。以各信号源重建后的误差和信号源之间的串扰来衡量恢复效果, 输入信噪比的变化范围是  $-5\text{dB} \sim +15\text{dB}$ 。实验结果证明 有噪 IFA 的分离效果明显地优于标准的 ICA 算法。例如 当  $\text{SNR}$  (信噪比) =  $15\text{dB}$  时在  $M = N = 3$  的情况下 有噪 IFA 算法的重建信号误差较 ICA 算法的低  $9\text{dB}$  串扰则低  $12\text{dB}$  当  $\text{SNR} = 10\text{dB}$  时 则分别低  $6\text{dB}$  和  $11\text{dB}$ 。LMS 和 MAP 两种恢复源信号的方法所得结果差异不大。

#### 7.11.4 无噪 IFA

##### 1. 为什么当 $N = 0$ 时有噪 IFA 退化为 PCA?

设有噪 IFA 生成模型中, 噪声  $\mathbf{N}$  的方差阵为  $\mathbf{A} = \eta \mathbf{I}$ ,  $\mathbf{I}$  为单位阵, 则可证明当  $\eta \rightarrow 0$  时, IFA  $\rightarrow$  PCA。首先 可以观察当  $\eta \rightarrow 0$  时有噪 ICA 计算中的一个关键概率  $\hat{p}(\mathbf{S} | \mathbf{q}, \mathbf{X}, \boldsymbol{\lambda}(k))$  (式 (7-207)) 的变化。其中的方差阵和均值向量的趋向如下。

$$\begin{aligned} \lim_{\eta \rightarrow 0} \boldsymbol{\Sigma}_{\mathbf{q}} &= \mathbf{0} \\ \lim_{\eta \rightarrow 0} \boldsymbol{\zeta}_{\mathbf{q}}(\mathbf{X}) &= (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{X} = \boldsymbol{\rho}(\mathbf{X}) \end{aligned}$$

这时  $\hat{p}(\mathbf{S} | \mathbf{q}, \mathbf{X}, \boldsymbol{\lambda}(k))$  退化为下列形式:

$$\lim_{\eta \rightarrow 0} \hat{p}(\mathbf{S} | \mathbf{q}, \mathbf{X}, \boldsymbol{\lambda}(k)) = \delta(\mathbf{S} - (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{X}) = \delta(\mathbf{S} - \boldsymbol{\rho}(\mathbf{X}))$$

$\delta(\cdot)$  是一个狄拉克  $\delta$  函数。可以看到 当噪声趋向于 0 时, 此条件概率中的条件  $\mathbf{q}$  不再起作用, 因此信号源 pdf 的 MOG 假设不再起作用 这时 EM 算法中的矩阵  $\mathbf{A}$  迭代计算结果将成为 PCA 的计算结果。这可以证明如下。当  $\eta \rightarrow 0$  时 式 (7-208) 将写成下列形式:

$$\begin{aligned} \langle \mathbf{S}^T | \mathbf{X}, k \rangle &= \boldsymbol{\rho}^T(\mathbf{X}) = (\mathbf{A}^T(k) \mathbf{A}(k))^{-1} \mathbf{A}^T(k) \mathbf{X}^T \\ \langle \mathbf{S} \mathbf{S}^T | \mathbf{X}, k \rangle &= \boldsymbol{\rho}(\mathbf{X}) \boldsymbol{\rho}^T(\mathbf{X}) \end{aligned}$$

则有

$$\begin{aligned}\langle \mathbf{X}(\mathbf{S} | \mathbf{X}, k) \rangle &= E_{\mathbf{X}}[\mathbf{X}((\mathbf{A}^T(k)\mathbf{A}(k))^{-1}\mathbf{A}^T(k)\mathbf{X})^T] \\ &= E_{\mathbf{X}}[\mathbf{X}\mathbf{X}^T]\mathbf{A}(k)(\mathbf{A}^T(k)\mathbf{A}(k))^{-1} \\ \langle \mathbf{S}\mathbf{S}^T | \mathbf{X}, k \rangle &= E_{\mathbf{X}}[(\mathbf{A}^T(k)\mathbf{A}(k))^{-1}\mathbf{A}^T(k)\mathbf{X}((\mathbf{A}^T(k)\mathbf{A}(k))^{-1}\mathbf{A}^T(k)\mathbf{X})^T] \\ &= (\mathbf{A}^T(k)\mathbf{A}(k))^{-1}\mathbf{A}^T(k)E_{\mathbf{X}}[\mathbf{X}\mathbf{X}^T]\mathbf{A}(k)(\mathbf{A}^T(k)\mathbf{A}(k))^{-1}\end{aligned}$$

将此二式代入式 (7-219) 并作简单整理, 即得到有噪 IFA 算法中当  $\eta \rightarrow 0$  时  $\mathbf{A}(k+1)$  的递推计算公式:

$$\mathbf{A}(k+1) = E_{\mathbf{X}}[\mathbf{X}\mathbf{X}^T]\mathbf{A}(k)[\mathbf{A}^T(k)E_{\mathbf{X}}[\mathbf{X}\mathbf{X}^T]\mathbf{A}(k)]^{-1}\mathbf{A}^T(k)\mathbf{A}(k) \quad (7-235)$$

如果  $\mathbf{A}(k)$  的  $N$  个  $M$  维列向量  $\mathbf{A}_i(k) = [a_{1i}(k), \dots, a_{Mi}(k)]^T, i = 1 \sim M (M \geq N)$  是相关阵  $\mathbf{R}_{\mathbf{X}\mathbf{X}} = E_{\mathbf{X}}[\mathbf{X}\mathbf{X}^T]$  前  $N$  个 (从大到小排列) 特征值相应的特征向量 (它们是归一正交向量), 则下列各方程成立:

$$\left. \begin{aligned}\mathbf{A}^T(k)\mathbf{R}_{\mathbf{X}\mathbf{X}}\mathbf{A}(k) &= \text{diag}\{\lambda_1, \dots, \lambda_N\}, \quad \mathbf{A}^T(k)\mathbf{A}(k) = \mathbf{I}_N \\ \mathbf{A}(k)\text{diag}\{\lambda_1, \dots, \lambda_N\}\mathbf{A}^T(k) &= \mathbf{R}_{\mathbf{X}\mathbf{X}}, \quad \mathbf{A}(k)\mathbf{A}^T(k) = \mathbf{I}_M\end{aligned}\right\} \quad (7-236)$$

其中  $\lambda_1, \dots, \lambda_N$  是  $\mathbf{R}_{\mathbf{X}\mathbf{X}}$  的前  $N$  个特征值,  $\mathbf{I}_N$  和  $\mathbf{I}_M$  分别是  $N \times N$  维和  $M \times M$  维单位阵。将式 (7-236) 代入式 (7-235) 即可得

$$\mathbf{A}(k+1) = \mathbf{A}(k)$$

这说明当  $\mathbf{A}(k)$  为 PCA 的解时有噪 IFA 的解将始终停留在这个解上, 因而不能实现源信号恢复。当噪声从零缓慢增加时有噪 IFA 的解将逐渐脱离 PCA 而转向正确的源信号恢复。因此有噪 IFA 只能用于噪声不是太小的场合。本节中将讨论无噪 IFA 算法。这种算法在  $N=0$  的条件下将 ICA 结合到采取 MOG 的 EM 算法中, 从而可能更高效地解决 BSS 问题。

## 2. 无噪 IFA 的目标函数与 EM 算法

与 7.11 节的前提相同 设有  $N$  个统计独立的随机变量  $S_i, i = 1 \sim N$  构成源信号向量  $\mathbf{S}, \mathbf{S} = [S_1, \dots, S_N]^T$ 。各  $S_i$  的 pdf 记为  $\hat{p}_i(S_i | \boldsymbol{\Omega}, \boldsymbol{\xi}_i)$  用式 (7-182) 的 MOG 函数表示。 $\mathbf{S}$  的 pdf 记为  $\hat{p}(\mathbf{S} | \boldsymbol{\Omega}, \boldsymbol{\xi})$  用式 (7-183) 表示, 并且可以用式 (7-184) ~ (7-187) 各式来作进一步描述。以上内容对于无噪 IFA 和有噪 IFA 是完全相同的。由于无噪 IFA 讨论  $N=0$  的情形  $\mathbf{X}$  与  $\mathbf{S}$  之间关系不能再用式 (7-188) 描述, 而应该是  $\mathbf{X} = \mathbf{A}\mathbf{S}$ 。其中观察向量  $\mathbf{X} = [x_1, \dots, x_M]^T (M \geq N)$ ,  $\mathbf{A}$  是  $M \times N$  维矩阵。这一小节讨论  $M=N$  的情况 ( $M > N$  的情况见 (3))。若  $\mathbf{A}$  满秩, 设  $\mathbf{A}^{-1} = \mathbf{W}, \mathbf{S} = \mathbf{W}\mathbf{X}$  则  $\mathbf{X}$  的模型生成概率记为  $\hat{p}(\mathbf{X} | \boldsymbol{\lambda})$  按照式 (7-15) 可用下式计算:

$$\hat{p}(\mathbf{X} | \boldsymbol{\lambda}) = |\det \mathbf{W}| \hat{p}(\mathbf{S} | \boldsymbol{\Omega}, \boldsymbol{\xi}) \Big|_{\mathbf{S}=\mathbf{W}\mathbf{X}} \quad (7-237)$$

其中  $\boldsymbol{\lambda} = \{\mathbf{W}, \boldsymbol{\Omega}, \boldsymbol{\xi}\}$ 。无噪声 IFA 算法的第一步是求最佳  $\boldsymbol{\lambda}$  使  $\hat{p}(\mathbf{X} | \boldsymbol{\lambda})$  与  $\mathbf{X}$  的真实 pdf  $p(\mathbf{X})$  最接近, 然后用所得最佳参数从  $\mathbf{X}$  求  $\mathbf{S}$ 。与有噪 IFA 情况同,  $\hat{p}(\mathbf{X} | \boldsymbol{\lambda})$  与  $p(\mathbf{X})$  之间的差距用 KL 距离衡量, 从而得到式 (7-194) 表示的目标函数  $L(\boldsymbol{\lambda})$ 。接着按迭代节拍  $k$  用 EM 算法对  $\boldsymbol{\lambda}$  进行迭代计算, 以求得使  $L(\boldsymbol{\lambda})$  达到极小值的  $\boldsymbol{\lambda}$ 。

### (1) E-步

设节拍  $k$  已求得  $\boldsymbol{\lambda}(k)$  在节拍  $k+1$  时应求  $\boldsymbol{\lambda}(k+1)$  使  $L(\boldsymbol{\lambda}(k+1))$  尽量小。为此将



式(7-237)代入式(7-194),得到

$$L(\lambda(k+1)) = -E_x \left[ \ln |\det \mathbf{W}(k+1)| + \ln \hat{p}(\mathbf{S} | \boldsymbol{\Omega}(k+1), \boldsymbol{\xi}(k+1)) \right]_{\mathbf{S}=\mathbf{W}(k+1)\mathbf{X}} \quad (7-238)$$

在下列推导中,为简化书写而暂时略去  $E_x[\cdot]$  运算符,待最后补入。再次用 Jenssen 不等式,得到

$$\begin{aligned} L(\lambda(k+1)) &= -\ln |\det \mathbf{W}(k+1)| - \ln \hat{p}(\mathbf{S} | \boldsymbol{\Omega}(k+1), \boldsymbol{\xi}(k+1)) \Big|_{\mathbf{S}=\mathbf{W}(k+1)\mathbf{X}} \\ &\leq -\ln |\det \mathbf{W}(k+1)| - \sum_q [\hat{p}(\mathbf{q} | \mathbf{S}, \boldsymbol{\Omega}(k), \boldsymbol{\xi}_q(k)) \Big|_{\mathbf{S}=\mathbf{W}(k)\mathbf{X}}] \cdot \\ &\quad \ln \frac{\hat{p}(\mathbf{q}, \mathbf{S} | \boldsymbol{\Omega}(k+1), \boldsymbol{\xi}(k+1)) \Big|_{\mathbf{S}=\mathbf{W}(k+1)\mathbf{X}}}{\hat{p}(\mathbf{q} | \mathbf{S}, \boldsymbol{\Omega}(k), \boldsymbol{\xi}_q(k)) \Big|_{\mathbf{S}=\mathbf{W}(k)\mathbf{X}}} \\ &= \mathcal{F}(\lambda(k+1), \lambda(k)) \end{aligned} \quad (7-239)$$

注意,这里和有噪 IFA 算法中导出的式(7-196)有一项重要区别。彼处在已知  $\mathbf{X}$  的条件下,  $\mathbf{S}$  仍属随机变量,所以 Jensen 不等式的右侧既含对于随机数  $\mathbf{q}$  的取和又含对随机变量  $\mathbf{S}$  的积分。而此处处在已知  $\mathbf{X}$  的条件下,  $\mathbf{S}$  将只是  $\mathbf{W}(k+1)$  的函数而不是随机变量,所以 Jensen 不等式右侧不含对  $\mathbf{S}$  的积分。如果略去与  $\lambda(k+1)$  无关的  $H_x(\mathbf{q} | \mathbf{S}, \boldsymbol{\Omega}(k), \boldsymbol{\xi}_q(k)) = -\sum_q \hat{p}(\mathbf{q} | \mathbf{S}, \boldsymbol{\Omega}(k), \boldsymbol{\xi}_q(k)) \ln \hat{p}(\mathbf{q} | \mathbf{S}, \boldsymbol{\Omega}(k), \boldsymbol{\xi}_q(k)) \Big|_{\mathbf{S}=\mathbf{W}(k)\mathbf{X}}$  则  $\mathcal{F}(\cdot, \cdot)$  可表示为

$$\begin{aligned} \mathcal{F}(\lambda(k+1), \lambda(k)) &= -\ln |\det \mathbf{W}(k+1)| - \sum_q [\hat{p}(\mathbf{q} | \mathbf{S}, \boldsymbol{\Omega}(k), \boldsymbol{\xi}_q(k)) \Big|_{\mathbf{S}=\mathbf{W}(k)\mathbf{X}}] \cdot \\ &\quad \ln \hat{p}(\mathbf{q}, \mathbf{S} | \boldsymbol{\Omega}(k+1), \boldsymbol{\xi}_q(k+1)) \Big|_{\mathbf{S}=\mathbf{W}(k+1)\mathbf{X}} \end{aligned} \quad (7-240)$$

(2) M-步

基于式(7-240)在已知  $\lambda(k) = \{\mathbf{W}(k), \boldsymbol{\Omega}(k), \boldsymbol{\xi}(k)\}$  的基础上求  $\lambda(k+1) = \{\mathbf{W}(k+1), \boldsymbol{\Omega}(k+1), \boldsymbol{\xi}(k+1)\}$  使  $\mathcal{F}(\cdot, \cdot)$  达到最小。引用式(7-183)、式(7-184)、式(7-186)、式(7-187)式(7-240)右侧第2项可以写成

$$\begin{aligned} & - \sum_{i=1}^N \sum_{q_i=1}^{Q_i} \left\{ \hat{p}_i(q_i | S_i, \lambda(k)) \left[ \ln \hat{p}_i(q_i | \boldsymbol{\Omega}(k+1)) + \right. \right. \\ & \quad \left. \left. \ln \hat{p}_i(S_i | q_i, \boldsymbol{\xi}_i(k+1)) \Big|_{\mathbf{S}=\mathbf{W}(k+1)\mathbf{X}} \right] \right\} \hat{p}_i(q_i | S_i, \lambda(k)) \\ &= \frac{\hat{p}_i(q_i | \boldsymbol{\Omega}(k)) \hat{p}_i(S_i | q_i, \boldsymbol{\xi}_i(k)) \Big|_{\mathbf{S}=\mathbf{W}(k)\mathbf{X}}}{\sum_{q'_i=1}^{Q_i} \hat{p}_i(S_i | q'_i, \boldsymbol{\xi}_i(k))} \quad (7-241) \\ & \left. \begin{aligned} \hat{p}_i(q_i | \boldsymbol{\Omega}(k+1)) &= \omega_{q_i}(k+1), \hat{p}_i(q_i | \boldsymbol{\Omega}(k)) = \omega_{q_i}(k) \\ \hat{p}_i(S_i | q_i, \boldsymbol{\xi}_i(k+1)) &= \frac{1}{\sqrt{2\pi}\sigma_{q_i}(k+1)} \exp \left\{ -\frac{1}{2} \left( \frac{S_i - \mu_{q_i}(k+1)}{\sigma_{q_i}(k+1)} \right)^2 \right\} \\ \hat{p}_i(S_i | q_i, \boldsymbol{\xi}_i(k)) &= \frac{1}{\sqrt{2\pi}\sigma_{q_i}(k)} \exp \left\{ -\frac{1}{2} \left( \frac{S_i - \mu_{q_i}(k)}{\sigma_{q_i}(k)} \right)^2 \right\} \end{aligned} \right\} \end{aligned}$$

将式(7-241)代入式(7-240)就得到了  $\mathcal{F}(\lambda(k+1), \lambda(k))$  的完整计算式。如果将此目标函数与式(7-45)给出的 ICA 目标函数  $\tilde{L}(\mathbf{W})$  进行对比(注意,ICA算法中的  $y_i(t)$  对应此

处的  $S_i$  式(7-45)中所作的批平均  $\frac{1}{T} \sum_{t=1}^T$  在式(7-240)和式(7-241)中暂时略去)就可以看到, 后者的惟一参变量是  $\mathbf{W}$  而前者还包含各  $\omega_{iq_i}(k+1)$ ,  $\mu_{iq_i}(k+1)$  和  $\sigma_{iq_i}(k+1)$  对于 ICA 而言各源 pdf,  $\hat{p}_i(y_i(t))$  被假设为已知的。

在无噪 IFA 算法中求最佳  $\mathbf{W}(k+1)$  和求最佳的诸  $\omega_{iq_i}(k+1)$ ,  $\mu_{iq_i}(k+1)$  和  $\sigma_{iq_i}(k+1)$  的方法有所区别。先讨论前者。如果在求  $\mathbf{W}(k+1)$  的过程中保持  $\boldsymbol{\Omega}(k+1)$  和各  $\xi_i(k+1)$  中的参数不变, 那么式(7-240)的  $\mathcal{F}(\boldsymbol{\lambda}(k+1), \boldsymbol{\lambda}(k))$  和式(7-45)的  $L(\mathbf{W})$  作为  $\mathbf{W}(k+1)$  系数 ——  $\mathbf{W}(k+1) \sim \mathbf{W}, \hat{p}_i(q_i | S_i, \boldsymbol{\lambda}(k)) [\ln \hat{p}_i(S_i | q_i, \xi_i(k+1)) |_{s=\mathbf{W}(k+1)\mathbf{X}}] \sim \ln \hat{p}_i(y_i(t)) |_{Y(t)=\mathbf{W}\mathbf{X}}, \mathbf{S} \sim \mathbf{Y}(t)$  这样在求最佳  $\mathbf{W}(k+1)$  时可以直接引用 ICA 算法中相对梯度迭代计算公式(式(7-59))来求  $\mathbf{W}(k+1)$ 。

$$\left. \begin{aligned} \mathbf{W}(k+1) &= \mathbf{W}(k) + \Delta \mathbf{W}(k) \\ \Delta \mathbf{W}(k) &= \alpha(k) [\mathbf{I} - \langle \boldsymbol{\varphi}(\mathbf{S}) \mathbf{S}^T \rangle] \mathbf{W}(k) \\ \boldsymbol{\varphi}(\mathbf{S}) &= [\varphi_1(S_1), \dots, \varphi_N(S_N)]^T, \varphi_i(S_i) = \sum_{q_i=1}^{Q_i} \hat{p}_i(q_i | S_i, \boldsymbol{\lambda}(k)) \cdot \\ &\quad \frac{S_i - \mu_{iq_i}(k+1)}{\sigma_{iq_i}^2(k+1)}, \quad i = 1 \sim N \\ \langle \boldsymbol{\varphi}(\mathbf{S}) \mathbf{S}^T \rangle &= E_{\mathbf{X}}[\boldsymbol{\varphi}(\mathbf{S}) \mathbf{S}^T |_{s=\mathbf{W}(k)\mathbf{X}}] \end{aligned} \right\} \quad (7-242)$$

在实际运行时集合平均用批平均代替。再讨论  $\omega_{iq_i}(k+1)$ ,  $\mu_{iq_i}(k+1)$  和  $\sigma_{iq_i}(k+1)$  的计算。设  $\mathbf{W}(k+1)$  保持不变, 则可以仿照有噪 IFA 中求这些参数的算法, 令  $\mathcal{F}(\boldsymbol{\lambda}(k+1), \boldsymbol{\lambda}(k))$  对这些参数的导数为 0 即可以求得(参照式(7-223)、式(7-224)和式(7-225))

$$\mu_{iq_i}(k+1) = \frac{E_{\mathbf{X}}\{[\hat{p}_i(q_i | S_i, \boldsymbol{\lambda}(k)) |_{s=\mathbf{W}(k)\mathbf{X}}][S_i |_{s=\mathbf{W}(k+1)\mathbf{X}}]\}}{E_{\mathbf{X}}[\hat{p}_i(q_i | S_i, \boldsymbol{\lambda}(k)) |_{s=\mathbf{W}(k)\mathbf{X}}]}, \quad i = 1 \sim N, q_i = 1 \sim Q_i \quad (7-243)$$

$$\sigma_{iq_i}^2(k+1) = \frac{E_{\mathbf{X}}\{[\hat{p}_i(q_i | S_i, \boldsymbol{\lambda}(k)) |_{s=\mathbf{W}(k)\mathbf{X}}][S_i^2 |_{s=\mathbf{W}(k+1)\mathbf{X}}]\}}{E_{\mathbf{X}}[\hat{p}_i(q_i | S_i, \boldsymbol{\lambda}(k)) |_{s=\mathbf{W}(k)\mathbf{X}}]}, \quad i = 1 \sim N, q_i = 1 \sim Q_i \quad (7-244)$$

$$\omega_{iq_i}(k+1) = E_{\mathbf{X}}[\hat{p}_i(q_i | S_i, \boldsymbol{\lambda}(k)) |_{s=\mathbf{W}(k)\mathbf{X}}], \quad i = 1 \sim N, q_i = 1 \sim Q_i \quad (7-245)$$

同样在实际计算中  $E_{\mathbf{X}}[\cdot]$  用批平均取代。

一个可执行的无噪 IFA-EM 算法按节拍  $k$  进行迭代计算。设节拍  $k$  时的参数已求得为  $\boldsymbol{\lambda}(k) = \{\mathbf{W}(k), \boldsymbol{\Omega}(k), \xi(k)\}$  则按下列方法求  $\boldsymbol{\lambda}(k+1) = \{\mathbf{W}(k+1), \boldsymbol{\Omega}(k+1), \xi(k+1)\}$ 。在此过程中执行一个小的循环迭代计算。设小循环的节拍为  $l$  参数用  $\boldsymbol{\lambda}(l)$  表示  $\boldsymbol{\lambda}(l) = \{\hat{\mathbf{W}}(l), \hat{\boldsymbol{\Omega}}(l), \hat{\xi}(l)\}$  则计算步序如下。

(1) 计算  $\hat{p}_i(q_i | S_i, \boldsymbol{\lambda}(k))$  (这一步计算完全用节拍  $k$  的参数 见式(7-241) 在小循环中这一参数保持不变) ;

(2) 令  $\boldsymbol{\lambda}(l) |_{l=0} = \boldsymbol{\lambda}(k)$  即  $\mathbf{W}(0) = \mathbf{W}(k), \boldsymbol{\Omega}(0) = \boldsymbol{\Omega}(k), \dots$  ;

(3) 令  $l = 0$  ;

(4) 计算  $S(l) = W(l)X$ ;

(5) 用式 (7-243)、式 (7-244), 计算各  $\hat{\mu}_{q_i}(l+1), \hat{\sigma}_{q_i}^2(l+1)$  (注意, 此二式中的  $W(k+1)$  现在应该用  $\hat{W}(l)$  取代, 其他不变);

(6) 用式 (7-242) 计算  $\hat{W}(l+1)$  (其中所用的  $\hat{S} = \hat{S}(l), \mu_{q_i}(k+1) = \hat{\mu}_{q_i}(l+1), \sigma_{q_i}^2(k+1) = \hat{\sigma}_{q_i}^2(l+1)$ );

(7) 判断是否收敛 (若  $\hat{\lambda}(l+1)$  与  $\lambda(l)$  之差异小于一个预定的阈值, 则收敛; 反之则否) 若收敛, 转 (8), 反之, 令  $l = l+1$ , 转 (4) 设收敛时的小循环节拍为  $l = L$ ;

(8) 用式 (7-245) 计算各  $\omega_{q_i}(k+1)$  (其中令  $W(k) = \hat{W}(L), \xi(k) = \hat{\xi}(L), \Omega(k)$  保持不变);

(9) 令  $W(k+1) = \hat{W}(L), \mu_{q_i}(k+1) = \hat{\mu}_{q_i}(L), \sigma_{q_i}^2(k+1) = \hat{\sigma}_{q_i}^2(L)$ ;

(10) 结束。

### 3. 归一化和 $M > N$ 情况的处理

与有噪 IFA 算法相同, 按节拍  $k$  完成一次迭代计算后进行一轮归一化处理可以加快收敛速度。归一化计算公式仍可采取式 (7-227) 和式 (7-228)。

对于  $M > N$  的情况, 可处理如下。设  $P$  是观察向量  $X$  相关阵  $R_{XX} = E[XX^T]$  的特征阵, 即下列方程成立:

$$P^T R_{XX} P = \text{diag}[\lambda_1, \dots, \lambda_M] \quad (7-246)$$

设  $\lambda_1, \dots, \lambda_M$  是按从大到小次序排列的特征值, 则  $P$  的从左至右第 1 至第  $M$  个列向量为与其相应的特征向量。设  $P$  的第 1 至第  $N$  列向量构成一个  $M \times N$  维矩阵  $P_1$ , 则

$$P_1^T R_{XX} P_1 = \text{diag}[\lambda_1, \dots, \lambda_N] \quad (7-247)$$

则可以用  $P_1^T$  将  $M$  维向量  $X$  转换成  $N$  维向量  $\tilde{X}$ ,

$$\tilde{X} = P_1^T X \quad (7-248)$$

然后, 可以用上述的算法求得矩阵  $\tilde{W}$  实现所需的源信号分离, 即

$$S = \tilde{W} \tilde{X} \quad (7-249)$$

由于  $S$  和  $\tilde{X}$  都是  $N$  维向量, 用  $M = N$  的无噪 IFA 求分离阵  $\tilde{W}$  是可行的。

总结以上讨论, 可以得到用无噪 IFA 算法从观察向量  $X$  恢复源信号向量  $S$  的方法为:

(1) 若  $M = N$ , 由  $X$  求出分离阵  $W$ , 则  $S$  用下式求:

$$S = WX$$

(2) 若  $M > N$ , 由  $X$  求其特征阵  $P$  这一部分任务由 PCA 完成。 $P$  的自左至右前  $N$  列向量构成与前  $N$  个最大特征值相应的矩阵  $P_1$ 。令  $\tilde{X} = P_1^T X$ 。针对  $\tilde{X}$  求出分离阵  $\tilde{W}$ , 则  $S$  用下式求:

$$S = \tilde{W} P_1^T X \quad (7-250)$$

## 4. GEM 算法

7.11.4 小节的 2 中讨论的算法是一种标准 EM 算法, 即每一迭代节拍  $k$  中都是在先固定  $\omega_{q_i}(k)$  的条件下交替进行  $W$  和  $\{\mu_{q_i}, \sigma_{q_i}^2\}$  的计算直至收敛, 最后算出  $\omega_{q_i}(k+1)$ 。文献 [43] 提出了一种广义 EM 算法, 即 GEM, 其中  $\omega_{q_i}$  的调整与其他参数的调整不严格区分

开。下面介绍两种 GEM 算法。

#### (1) 追逐 (chase) 算法

与 7.11.4 小节的 2 所述的标准 EM 算法基本相同，区别仅在于每个节拍  $k$  的小循环中只进行一次  $\mu_{iq_i}, \sigma_{iq_i}^2$  和  $W$  调整后，立即进行  $\omega_{iq_i}$  调整（即  $L \equiv 1$ ）。实际上可以按下列两种方式交替调整 3 组参数。第一种，调  $W$ ——调  $\{\mu_{iq_i}, \sigma_{iq_i}^2\}$ ——调  $\omega_{iq_i}$ 。第二种，调  $\{\mu_{iq_i}, \sigma_{iq_i}^2\}$ ——调  $W$ ——调  $\omega_{iq_i}$ 。其中每一步只进行一次调整计算且使用前一至三步调整中所得到的参数作为本步调整的前提参数（即  $\lambda(k)$ ）。

#### (2) 跷跷板 (seesaw) 算法

此算法用以下两段计算交替进行来构成。第一段，固定  $W$  交替调整  $\{\mu_{iq_i}, \sigma_{iq_i}^2\}$  和  $\omega_{iq_i}$  共  $L$  次， $L$  可设为一固定数或达到收敛时的次数。第二段，固定  $\{\mu_{iq_i}, \sigma_{iq_i}^2\}$  交替调整  $W$  和  $\omega_{iq_i}$  共  $L$  次， $L$  为固定或收敛次数。

模拟实验结果表明，大多数情况下这两种 GEM 算法的效果皆优于标准 GEM 算法。

#### 5. 无噪 IFA 的模拟实验结果<sup>[41]</sup>

信号源取为长度 5 秒的语音、音乐和合成信号各一段，实验条件是  $M = N = 3$ ； $Q_i = 3, \forall i; a(k) \equiv 0.05$ ；混合阵  $A$  为  $3 \times 3$  维随机阵。算法收敛标准是：对于分离阵  $W$  检验  $WA$  与  $I$  的差距。对于各 pdf 参数检验模型 pdf 和真实 pdf 之间的 KL 距离。实验结果是，chase 算法经过约 500 次迭代，seesaw 算法经过约 300 次迭代，则可使上述两项差距减至接近于 0。

## 7.12 ICA 和 IFA 的实际应用和待解决问题

下面列出 ICA 或 IFA 的若干实际应用，这里只列出有关的文献，从中可以找到相应的模拟实验结果。

(1) 鸡尾酒问题 (10 个说话人的话语分离)<sup>[2]</sup>。

(2) 医学：包括心电图 (ECG)<sup>[48]</sup> 和脑电图 (EEG) 的信号分离问题<sup>[46,5]</sup> 还有功能磁共振图像 (fMRI) 数据分析<sup>[47]</sup> 以及听觉信号分析<sup>[59]</sup>。

(3) 雷达和声纳：包括阵列信号处理、干扰抑制、源定位等<sup>[6,49,7]</sup>

(4) 通信：包括信道均衡，特别是移动通信中，于存在延时及多径传输情况下多用户的分离<sup>[50,51,16,53~55]</sup>

(5) 图像和语音信号处理：如图像增强和恢复、语音增强和恢复<sup>[2,52,56,58,5]</sup>

(6) 地球物理信号处理：探油、地震信号解卷等<sup>[2]</sup>

(7) 财金领域中的应用：如股价预测<sup>[57,13]</sup>

(8) 数据分析和压缩：如果  $M \gg N$ ，则能用少量的源信号表示大量观察信号，从而实现数据压缩<sup>[6]</sup>。

当 ICA 和 IFA 用于 BSS 和 BDC 时，尚有大量的理论和实际问题有待解决，下面只列举其中的一部分。

(1) 原信号 pdf 的学习。

(2) 盲解卷问题。

- (3) 全局收敛性问题。
- (4) 多维 ICA 问题 ( 源信号向量各分量分成若干组, 组内各分量存在相关性, 各组之间统计独立 )
- (5) 噪声存在时更有效的解法。
- (6) 如何更有效地利用各种先验知识。
- (7) 如何解  $M < N$  的情况 ( EEG 信号处理中即存在此处情况 ) 。
- (8) 当迭加噪声为非高斯的或脉冲噪声时, 如何准确估计源信号的个数。
- (9) 非线性混合的情形如何求解。
- (10) 非平稳情况下如何提高跟踪能力以及如何提高解的鲁棒性。
- (11) 如何与 NN 特别是 RNN 更好地相结合 ( 这部分内容文献 [8] 有详论 ) 。

## 参考文献

- [1] Cao X-R, Liu R-W. General approach to blind source separation. *IEEE Trans. Signal Processing*, April 1996, 78: 753 ~ 766
- [2] Bell Anthony J, Sejnowski Terrence J. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 1995, 7(6): 1004 ~ 1034
- [3] Lathauwer L De, et al. Fetal electrocardiogram extraction by source subspace separation. In: *Proc. HOS'95. Aiguablava, Spain; June 1995.* 134 ~ 138
- [4] Makeig S, et al. Independent component analysis in electroencephalographic data. In: *Advances in Neural Information Processing Systems*, 8, Denver, CO: MIT Press. 1996. 145 ~ 151
- [5] Lee Te-Won, et al. Independent component analysis using an extended infomax algorithm for mixed subgaussian and supergaussian sources. *Neural Computation*, 1999, 11: 417 ~ 441
- [6] Comon Pierre. Independent component analysis, a new concept. *Signal Processing*, 1994, 36: 287 ~ 314
- [7] Papadias C B, Paulraj A. A constant modulus algorithm for multi-user signal separation in presence of delay spread using antenna arrays. *IEEE Signal Processing Lett.*, June 1997, 4: 178 ~ 181
- [8] Amari S-I, Cichocki A. Adaptive blind signal processing-neural network approaches. *Proceedings of the IEEE*, Oct. 1998, 86(10): 2026 ~ 2046
- [9] Anand K, et al. Blind separation of multiple co-channel BPSK signals arriving at an antenna array. *IEEE Signal Processing Lett.*, Sept. 1995, 2: 176 ~ 178
- [10] Swindlehurst A, et al. Some experiments with array data collected in actual urban and suburban environments. In: *IEEE Workshop on Signal Processing Advances in Wireless Communications.* Apr. 1997. 301 ~ 304
- [11] Karhunen J, et al. Applications of neural blind separation to signal and image processing. In: *Proc. ICASSP.* 1997. 1: 131 ~ 134
- [12] Pearlmutter B A, Parra L C. A context-sensitive-generalization of ICA. In: *Proc. Int. Conf. Neural Information Processing, ICONIP'96. Hong Kong; Sep. 1996.* 151 ~ 157
- [13] Baram Y, Roth Z. Density shaping using neural Networks. In: *Computational Intelligence for Financial Engineering.* New York City: IEEE Press, 1995
- [14] Flandrin P, Michel O. Chaotic signal analysis and higher order statistics. In: *Conf. EUSIPCO.* Brussels; 1992. 179 ~ 182
- [15] Jutten C, Herault J. Blind separation of sources. Part I: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, July, 1991, 24(1): 1 ~ 10
- [16] Cardoso J-F, Laheld B H. Equivariant adaptive source separation. *IEEE Trans. Signal Processing*, Dec. 1996, 44(10): 3017 ~ 3030
- [17] Pham D T, Garat P. Blind separation of mixture of independent sources through a quasi-maximum likelihood approach. *IEEE Trans. Signal Processing*, July 1997, 45(7): 1712 ~ 1725
- [18] Cardoso J-F. Blind signal separation: statistical principles. *Proceedings of the IEEE*, Oct. 1998, 86(10): 2009 ~ 2025
- [19] Girolami M. An alternative perspective on adaptive independent component analysis algorithms.

Neural Comutation, 1998, 10: 2103 ~ 2114

- [20] Cardoso J-F. Infomax and maximum likelihood for blind source separation. *IEEE Signal Processing Lett.*, Apr. 1997, 4: 109 ~ 111
- [21] Yang H H, et al. Adaptive on-line learning algorithms for blind separation—Maximum entropy and minimum mutual information. *Neural Computation*, 1997, 7(9): 1457 ~ 1482
- [22] Amari Shun-ichi. Natural gradient works efficiently in learning. *Neural Computation*, 1998, 10: 251 ~ 276
- [23] Amari S, et al. A new learning algorithm for blind signal separation. In: *NIPS'95*, 8. Cambridge, MA; MIT Press, 1996
- [24] Cichocki A, et al. Self-adaptive neural networks for blind separation of sources. In: *Proc. 1996 IEEE Int. Symp. Circuits and Systems*. May 1996. 2, 157 ~ 160
- [25] Cichocki A, et al. On-line adaptive algorithms in nonstationary environments using a modified conjugate gradient approach. In: *Proc. IEEE Workshop Neural Networks for Signal Processing*. Sept. 1997. 316 ~ 325
- [26] Amari S, et al. Stability analysis of adaptive blind source separation. *Neural Networks*, 1997, 10(8): 1345 ~ 1351
- [27] Girolami M, Fyfe C. Kurtosis extrema and identification of independent components; a neural network approach. In: *Proc. ICASSP-97*. Apr. 1997. 4: 3329 ~ 3333
- [28] Loève M. *Probability theory*. Princeton, NJ: Van Nostrand, 1963
- [29] Delfosse N, Loubaton P. Adaptive blind separation of independent sources; a deflation approach. *Signal Processing*, 1995, 45: 59 ~ 83
- [30] Cichocki A, et al. Sequential blind signal extraction in order specified by stochastic properties. *Electron. Lett.*, Jan. 1997, 33(1): 64 ~ 65
- [31] Cichocki A, et al. Neural network models for blind separation of time delayed and convolved signals. *Japanese IEICE Trans. Fundamentals*, Sep. , 1997, E-82-A(9): 1057 ~ 1062
- [32] Haykin S, ed. *Blind deconvolution*. Englewood Cliffs, NJ: Prentice Hall, 1994
- [33] Inouye Y, Sato T. On-line algorithms for blind deconvolution of multichannel linear time-invariant systems. In: *Proc. IEEE Signal Processing Workshop on Higher-Order Statistics*. 1997. 204 ~ 208
- [34] Lee T W, et al. Blind separation of delayed and convolved sources. *Advances in Neural Information Processing Systems 9*, Cambridge, MA: MIT Press, 1997. 758 ~ 764
- [35] Douglas S C, Cichocki A. Neural networks for blind decorrelation of Signals. *IEEE Trans. Signal Processing*, Nov. 1997, 45: 2829 ~ 2842
- [36] Douglas S C, et al. Quasi-Newton filtered-error and filtered-regressor algorithms for adaptive equalization and deconvolution. In: *Proc. IEEE Workshop Signal Processing and Advances in Wireless Communications*. Paris; April, 1997. 109 ~ 112
- [37] Amari S, et al. Multichannel blind deconvolution and equalization using the natural gradient. In: *Proc. IEEE Workshop Signal Processing and Advances in Wireless Communications*. Paris; April, 1997. 101 ~ 104
- [38] Attias H, Schreiner C E. Blind source separation and deconvolution; the dynamic component analysis algorithm. *Neural Computation*, 1998, 10: 1373 ~ 1424
- [39] Everitt B S. *An introduction to latent variable models*. London; Chapman and Hall, 1984

- [40] Attias H. Independent factor analysis. *Neural Computation*, 1999, 11; 803 ~ 851
- [41] Roweis S. A unifying review of linear Gaussian models. *Neural Computation*, 1999, 11; 305 ~ 345
- [42] Neal R M, Hinton G E. A view of the EM algorithm that justifies incremental, sparse and other variants. In: Jordan M I, ed. *Learning in graphical models*. Dordrecht, MA: Kluwer, 1998. 355 ~ 368
- [43] Cover T M, Tomas J A. *Elements of information theory*. New York: Wiley, 1991
- [44] Roweis S. EM algorithms for PCA and SPCA. In: Kearns M, et al., ed. *Advances in Neural Information Processing Systems 10*. Cambridge, MA: MIT Press, 1998. 626 ~ 632
- [45] Makeig S, et al. Independent component analysis in electro-encephalographic data. In: Mözer M, et al., ed. *Advances in Neuro Information Processing System*, 8. Cambridge, MA: MIT Press, 1996. 145 ~ 151
- [46] Mckeown M, et al. Spatially independent activity patterns in functional magnetic resonance imaging data during the stroop Color-naming task. *Proceedings of the National Academy of Sciences*, 1998, 803 ~ 810
- [47] Lathauwer L De, et al. Fetal electrocardiogram extraction by source subspace separation. In: *Proc. HOS'95*. Aiguablava, Spain; June 1995. 134 ~ 138
- [48] Desodt G, Muller D. Complex independent component analysis applied to the separation of radar signals. In: *Proc. EUSIPCO Conf.*. Barcelona, Amsterdam; Elsevier, 1990. 665 ~ 668
- [49] Papadias C B, Paulraj A. A constant modulus algorithm for multi-user signal separation in presence of delay spread using antenna arrays. *IEEE Signal Processing Lett.*, June 1997, 4; 178 ~ 181
- [50] Swindlehurst A, et al. Some experiments with array data collected in actual urban and suburban environments. In: *IEEE Workshop on Signal Processing Advances in Wireless Communications*. Apr. 1997. 301 ~ 304
- [51] Cichocki A, et al. Neural network approach to blind separation and enhancement of images. *Signal Processing VIII, Theories and Applications*, Sept. 1996, 1; 579 ~ 582
- [52] Anand K, et al. Blind separation of multiple co-channel BPSK signals arriving at an antenna array. *IEEE Signal Processing Lett.*, Sept. 1995, 2; 176 ~ 178
- [53] Tugnait J K. Blind equalization and channel estimation for multiple-input multiple-output communication systems. In: *Proc. IEEE Int. Conf. Acoustics, Speech Signal Processing*. Atlanta, GA; May 1996, 5. 2443 ~ 2446
- [54] Veen A-J Van der, et al. A subspace approach to blind space-time signal processing for wireless communication systems. *IEEE Trans. Signal Processing*, Jan. 1997, 45; 173 ~ 190
- [55] Karhunen J, et al. Applications of neural blind separation to signal and image processing. In: *Proc. ICASSP*. 1997, 1; 131 ~ 134
- [56] Back A, Weigend A. A first application of independent component analysis to extracting structure from stock returns. *Int. J. Neural Syst.*, Aug. 1997, 8(4)
- [57] Bell A J, et al. Edges are the "independent components" of natural scenes. In: *Advances in Neural Information processing Systems*. Denver, CO: MIT Press, 1996, 9
- [58] Makeig S, et al. Blind Separation of event-related brain response into spatial independent components. In: *Proceedings of the National Academy of Sciences*, 94. 1997. 10979 ~ 10984



## 缩 略 语

<b>AI</b>	artificial intelligence	人工智能	<b>DEFK</b>	decoupled EFK	解耦 EFK
<b>ANN</b>	artificial neural network	人工神经网络	<b>DCA</b>	dynamic component analysis	动态分量分析
<b>ART</b>	adaptive-resonance theory	自适应谐振理论	<b>DCA</b>	dynamic channel assignment	动态信道分配
<b>Auto-AM</b>	auto-associative memory	自联想记忆	<b>EA</b>	evolutionary algorithm	进化算法
<b>ARMA</b>	autoregressive moving-average	自回归移动取平均	<b>EC</b>	evolutionary computation	进化计算
<b>BSP</b>	blind signal processing	盲信号处理	<b>EP</b>	evolutionary program	进化程序
<b>BSS</b>	blind signal separation	盲信号分离	<b>ERM</b>	empirical risk minimization	经验风险最小
<b>BAM</b>	bidirectional associative memory	双向联想记忆	<b>EKF</b>	extended Kalman filter	扩展卡尔曼滤波
<b>BSB</b>	brain-state-in-a-box	盒中脑状态	<b>EM</b>	expectation-maximization	期望最大化
<b>BSE</b>	blind signal extraction	盲信号抽取	<b>EHM</b>	extended Hopfield model	扩展 Hopfield 模型
<b>BP</b>	back propagation	反传	<b>ECG</b>	electrocardiogram	心电图
<b>CI</b>	computational intelligence	计算智能	<b>EEG</b>	electroencephalogram	脑电图
<b>CN</b>	control network	控制网络	<b>FIS</b>	fuzzy inference system	模糊推理系统
<b>CIMS</b>	computer-intergrated manufacturing system	计算机集成制造系统	<b>FNN</b>	fuzzy neural network	模糊神经网络
<b>CI</b>	context independent	与上下文无关	<b>FMRI</b>	functional magnetic resonance imaging	功能磁共振成像
<b>CAM</b>	content-addressed memory	按内容提取记忆	<b>FL</b>	fuzzy logics	模糊逻辑
<b>CSP</b>	car sequencing problem	车辆排序问题	<b>FLC</b>	fuzzy logic controller	模糊逻辑控制器
<b>CAP</b>	channel assignment problem	信道分配问题	<b>FCM</b>	fuzzy C-means	模糊 C-平均
<b>CAM</b>	competitive activation mechanism	竞争机制	<b>FA</b>	factor analysis	因子分析
<b>DM</b>	data mining	数据采掘	<b>FR</b>	filter-regressor	滤波-回归
<b>DB</b>	data base	数据库	<b>GA</b>	genetic algorithm	遗传算法
			<b>GEKF</b>	global EKF	扩展 Kalman 滤波
			<b>GPP</b>	graph partitioning problem	图分割问题

<b>GCN</b>	generalized clustering network	广义聚类网络	<b>MISO</b>	multi-input single-output	多输入单输出
<b>GMDH</b>	group method of data handling	集群数据处理技术	<b>MIMO</b>	multi-input multi-output	多输入多输出
<b>HMM</b>	hidden Markov model	隐含马尔可夫模型	<b>MLE</b>	maximum likelihood estimation	最大似然估计
<b>HNN</b>	Hopfield neural network	Hopfield 神经网络	<b>MMI</b>	maximum mutual information	最大互信息
<b>HM</b>	Hopfield model	Hopfield 模型	<b>MFT</b>	mean field theory	平均场理论
<b>Hetro-AM</b>	Hetro-associative memory	异联想记忆	<b>MFA</b>	mean field annealing	平均场退火
<b>HC</b>	hill-climbing	登山	<b>MMC</b>	modified mountain clustering	修正山形聚类
<b>HCM</b>	hard C-means	硬 C-平均	<b>MLD</b>	mean local density	平均局部密度
<b>ICA</b>	independent component analysis	独立分量分析	<b>ML</b>	maximum likelihood	最大似然
<b>IDN</b>	identification network	辨识网络	<b>MOG</b>	mixed of Gaussian function	混合高斯函数
<b>IFA</b>	independent factor analysis	独立因子分析	<b>NP</b>	nonpolynomial	非多项式的
<b>KDD</b>	knowledge discovery in database	数据库中发现知识	<b>NIIR</b>	nonlinear IIR	非线性无限冲击响应
<b>KNN</b>	K nearest neighbour	K 最近邻	<b>NMA</b>	nonlinear moving-average	非线性移动取平均
<b>KCN</b>	Kohonen clustering network	Kohonen 聚类网络	<b>NARMA</b>	nonlinear ARMA	非线性 ARMA
<b>LPN</b>	learning Petri network	学习 Petri 网络	<b>OLS</b>	orthogonal least square	正交最小平方
<b>LCD</b>	left context dependent	依赖于上文的	<b>PCA</b>	principle component analysis	主分量分析
<b>LVQ</b>	learning vector quantization	学习向量量化	<b>PCNN</b>	pulse coupled NN	脉冲耦合神经网络
<b>LPR</b>	linear programming relaxation	线性规划松弛	<b>PET</b>	positron emission tomograph	正电子辐射断层成像
<b>MLFN</b>	multilayer feedforward neural network	前向多层神经网络	<b>PPR</b>	projection pursuit regression	投影跟踪回归
<b>MBP</b>	modified back propagation	修正反传	<b>PNN</b>	probabilistic NN	概率神经网络
<b>MLP</b>	multilayer perceptron	多层感知器	<b>PLP</b>	perception weighted linear prediction	感觉加权线性预测
<b>MDL</b>	minimum description length	最小描述尺度	<b>pdf</b>	probability density function	概率密度函数
<b>MA</b>	moving average	移动取平均	<b>PDN</b>	postal delivery problem	邮递网络问题
			<b>RNN</b>	recurrent NN	递归神经网络
			<b>RO</b>	random optimization	随机优化
			<b>RCD</b>	right context dependent	依赖于下文的

<b>RLSE</b>	recurssive least square estimation	递归最小平方估计	<b>SIR</b>	signal to interference ratio	信号干扰比
<b>RC</b>	regularity criterion	规则化准则	<b>SCA</b>	static channel assignment	静态信道分配
<b>RBF</b>	radial basis function	径向基函数	<b>SUR</b>	simplified unbiased criterion	简化非偏准则
<b>SOM</b>	self-organized mapping	自组织映射	<b>ST</b>	search tree	搜索树
<b>SOFM</b>	self-organized feature mapping	自组织特征映射	<b>TSP</b>	travelling salesman problem	旅行商问题
<b>SA</b>	simulated annealing	模拟退火	<b>TDNN</b>	time delay NN	时延神经网络
<b>SVM</b>	support vector machine	支持向量机	<b>TDC</b>	tangent distance classifier	正切距离分类器
<b>SVD</b>	single valued decomposition	单值分解	<b>TSK</b>	Takagi-Sugeno-Kang model	TSK 模型
<b>SPET</b>	single photon emission tomograph	单光子辐射断层成像	<b>UC</b>	unbiased criterion	非偏准则
<b>STFT</b>	short time Fourier transformation	短时傅里叶变换	<b>VQ</b>	vector quantization	向量量化
<b>SISO</b>	single-input single-output	单输入单输出	<b>WVD</b>	Wigner-Viller distribution	维格纳-维拉分布

# 前言

人工神经网络以及与其密切相关的模糊推理（模糊逻辑、模糊集）、进化算法（本书主要讨论其中的遗传算法）和盲信号处理等构成了一门新学科——计算智能学——的核心内容。这门新学科的主要目标是实现用计算机来替代并且更好地完成人的各种智能工作。这是 21 世纪科技界面临的重大挑战之一，它不但对于信息科学技术而且对于经济、军事、工业生产和生物医学等领域将产生非常深远的影响。本书第 1 章为绪论 其余章节以人工神经网络为重点，分 3 个部分来介绍这一课题。第 1 部分从第 2 章至第 4 章，包括 3 种主要的人工神经网络：前向多层神经网络（含递归神经网络）、自组织神经网络和 Hopfield 神经网络。其中第一种研究得最为广泛、深入且应用面最宽，迄今仍是最重要的发展方向之一。第二种的学习算法与构成思路 and 第一种不同，由于它在模式识别、聚类和数据采掘等领域中表现出独特的优势，近年来发展很快，受重视程度日益提高。第三种是最早提出并促进人工神经网络快速发展的主要几种网络之一，其主要用途是优化和联想记忆。虽然其发展过程有一些曲折，但是近年来它的一些不足之处已全部或部分解决，目前仍然是一个重要的研究方向。第 2 部分为第 5 章和第 6 章，包括模糊神经网络和遗传算法在人工神经网络中的应用两项内容。人工神经网络的优点是具有强学习能力而缺点是不容易纳入人的推理知识，模糊推理系统的优缺点正好与其相反。二者相结合构成的模糊神经网络可以取长补短，因而有很大优势，近年来发展很快，在信号处理、控制、机器人等领域中起重要作用。遗传算法是一种并行逐代优化算法，在解决很多复杂优化问题时表现出独特的优势。而人工神经网络的参数和结构学习正是一个复杂优化问题，所以二者的结合将使人工神经网络学习效率有很大提高。第 3 部分为第 7 章，是盲信号处理，其中主要包含盲解卷和盲分离两大部分内容。所谓盲是指对于被处理的信号没有或只有很少先验知识的条件下，实现多个相加混合信号的分离或卷积信号的解卷。在军事、生物医学、声学 and 地球物理等许多领域中都需解决这类问题。这一课题是人工神经网络和统计信号处理两大学科相互结合的产物，近年来异军突起且成果斐然，已成为人工神经网络研究的一个新重点。

本书的写作可以追溯到 1989 年春季我们为清华大学电子工程等系博士生、硕士生开设的“人工神经网络”课程。1992 年在授课的基础上并结合我们自己的研究工作编写了一本教材——《人工神经网络》（北京 高等教育出版社，1992. 9）。该书的出版受到了国内、外读者的欢迎。十年来这一领域的研究与应用有了极大的进展而且产生了许多新的重要研究方向。这样，编写一本新书显得十分必要，国内很多同行及各界涉及这一领域的人士也非常关心这项工作的进展。从 1998 年动笔，历经四年完成了本书的写作。在写作过程中，我们想提到两位先贤的学术思想对于我们的影响。一位是曾在我校长期执教的陈寅恪教授（1890—1969）。他毕生提倡学术研究的“独立之精神、自由之思想”，并且身体

力行，已成为近百年来中国学术思想界的楷模。另一位是英国科学哲学家卡尔·波普尔（Karl Popper, 1902—1994），他的许多思想曾对人工神经网络的研究产生过深刻的影响。他的论述也对我们深有启发，特别是在科学研究和创新的过程中运用批评的精神、通过“证伪”引起科学进步的思想以及独立运用理智的必要性等，都值得我们反复思考。

关于本书，其初衷是作为一本博士生和硕士生的教材，但从完稿的篇幅和内容看已大大超出了这个目标，即使一门 64 学时的课程也很难将本书的全部内容讲完。因此在授课时只能讲授关键和重点部分，而较深入部分可作为自学或研究参考使用。本书的写作着眼于方便读者自学，即每一章的内容都是自给自足的，既包含每一研究课题的背景、发展历史、基本假设条件、研究思路、重要算法、重要结论、工程应用以及未来发展方向等，又包含有关各重要定理和算法的证明和推导。对于不给出证明的部分，则给出相应的参考文献。本书列出了五百余篇参考文献，虽然远不全面，但是关键文献以及关于未来发展方向论述的文献大体具备。

本书写作的分工是：

第 1、2、3、4、5、7 章由杨行峻撰写。第 6 章由郑君里撰写，陈文霞博士协助完成了第 6 章的部分编写工作。

本书的完成和出版得到了多方面的鼓励和帮助。这里要特别提出的是清华大学“985 教材基金”对本书编写给予的资助。对此我们表示由衷的感谢。最后，对长期关心本书出版的各界同行深致谢忱，并恳请读者对书中不足之处批评指正。

作者

2002 年 8 月于清华园